Desenvolvimento móvel: Uma solução Android completa com Rad Studio

Clube Delphi

DbExpress de A-Z
DataSnap em ação

Gerenciamento de Riscos
Seu projeto sob controle

Ano 12 :: Edição 143 :: R$ 11,90

DevMedia

GERÊNCIA DE CONFIGURAÇÃO

Versionamento com Delphi

Estilo Metro no Delphi
Conheça o
TMS Metro Controls

ER/Studio Business Architect XE 2
Modelagem UML
de forma integrada

ISSN 1517990-7

Somos tão apaixonados por tecnologia que o nome da empresa

diz tudo.

Porta 80 é o melhor que a Internet pode oferecer para sua empresa.

Já completamos 8 anos e estamos a caminho dos 80, junto com nossos clientes.

Adoramos tecnologia.
Somos uma equipe composta de gente que entende e gosta do que faz,

assim como você.

Estrutura

100% NACIONAL. Servidores de primeira linha, links de alta capacidade.

Suporte diferenciado

Treinamos nossa equipe para fazer mais e melhor. Muito além do esperado.

Serviços

Oferecemos a tecnologia mais moderna, serviços diferenciados e antenados com as suas necessidades.

1-to-1

Conhecemos nossos clientes. Atendemos cada necessidade de forma única. Conheça!

Porta 80
WEB HOSTING

Hospedagem | Cloud Computing | Dedicados | VoIP | Ecommerce | Aplicações | Streaming | Email corporativo

porta80.com.br | comercial@porta80.com.br | twitter.com/porta80

SP 4063-8616 | RJ 4063-5092 | MG 4063-8120 | DF 4063-7486

Sumário

Boas Práticas

Boa Prática

06 – Gerência de configuração – Versionamento
[*Paulo Roberto Quicoli*]

Delphi

Boa Ideia

14 – Estilo Metro no Delphi
[*Paulo Roberto Quicoli*]

Novidade

21 – ER/Studio Software Architect XE 2
[*Fabrício Hissao Kawata*]

Minicurso

28 – Desenvolvimento Android – Parte 3
[*Giuliano Scombatti Pinto*]

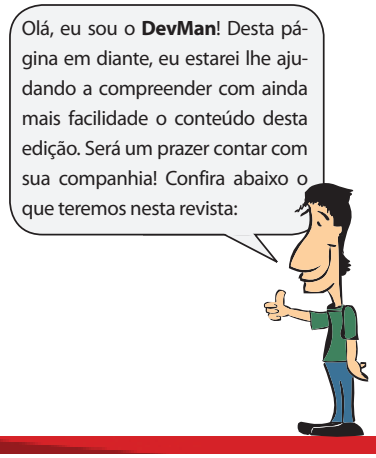
Easy, Minicurso

36 – dbExpress de A a Z - Parte 2
[*Alexandre Borges*]

Eng. de Software

Eng. de Software

44 – Gerenciamento de Riscos
[*Guinther Pauli*]



Dê seu feedback sobre esta edição!

A ClubeDelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre esta edição, artigo por artigo, através do link:
www.devmedia.com.br/clubedelphi/feedback

Clube Delphi

Ano 12 - 143ª Edição 2012 - ISSN 1517990-7 Impresso no Brasil

Corpo Editorial

Editor Geral

Paulo Quicoli
pauloquicoli@gmail.com

Editor Técnico

Daniel Sobrinho Laporte
daniel.laporte@gmail.com

Equipe Editorial

Guinther Pauli
guintherpauli@gmail.com

Fabrício Hissao Kawata
fabricio.kawata@bol.com.br

Giuliano Scombatti Pinto
giuliano@sygnux.com.br

Jornalista Responsável

Kaline Dolabella - JP24185

Atendimento ao Leitor

A DevMedia conta com um departamento exclusivo para o atendimento ao leitor. Se você tiver algum problema no recebimento do seu exemplar ou precisar de algum esclarecimento sobre assinaturas, exemplares anteriores, endereço de bancas de jornal, entre outros, entre em contato com:

www.devmedia.com.br/mancad
(21) 3382-5038

Publicidade

Para informações sobre veiculação de anúncio na revista ou no site e para fechar parcerias ou ações específicas de marketing com a DevMedia, entre em contato com:

publicidade@devmedia.com.br

Distribuição

FC Comercial e Distribuidora S.A
Rua Teodoro da Silva, 907
Grajaú - RJ - 206563-900

EDITORIAL

Ninguém faz nada sozinho nesse mundo, ainda mais quando o assunto é desenvolvimento de software. Existe uma prática que trata como controlar o desenvolvimento de um software para garantir sua evolução e qualidade ao longo do tempo e envolvendo várias pessoas no projeto. É a gerência de configuração. Com ela podemos ter um ambiente propício ao desenvolvimento concorrente de versões. Um de seus braços é o controle de versão, como gerir o acesso de um mesmo código a várias pessoas. É um desafio que é solucionado através do uso de gerenciadores de versão e neste artigo apresentado por mim conheceremos o Subversion e como o Delphi se integra a ele, mas além disso vamos entender a gerência de configuração em um contexto mais amplo e até mesmo ver um processo de versionamento comentado. Falando em "ao longo do tempo", todo software evolui, em especial o Windows. Assim como o Windows 95 quebrou um paradigma em relação ao Windows 3.1, o Windows 8 também quebrará. Temos novos recursos visuais (componentes, efeitos visuais e padronização) disponíveis que podem nos inspirar no desenvolvimento de nossos aplicativos. Esse visual é chamado de Metro e já está em uso em algumas plataformas Windows como Xbox Live e Windows Phone. O Delphi sempre contou com o apoio de várias empresas desenvolvedoras de componentes para entregar uma funcionalidade que agilizasse a criação de aplicações. Embora a Embarcadero ainda não tenha disponibilizado uma atualização da VCL para esse novo layout, a TMS não falhou e criou o TMS Metro Controls. Um conjunto de controles que permite o uso de alguns dos conceitos envolvidos nessa linguagem visual Metro. Eu trago um artigo sobre isso, mostrando como é possível revitalizar um simples aplicativo.

O Fabrício fala sobre o ER/Studio Software Architect XE 2 e como ele pode auxiliar na modelagem de um sistema utilizando a UML. Realizar essa modelagem é importante pois pode compartilhar entre a equipe um censo comum do que está sendo feito, além de servir como uma documentação montada em UML, um padrão indiscutível de modelagem de sistemas.

Continuando o tour sobre as funcionalidades do DbExpress, o Alexandre agora aborda o uso do DataSnap para geração de uma aplicação servidora que compartilha dados no formato JSON. Esse formato de transmissão de dados tem se popularizado pela Web, grandes sites como Facebook o utilizam justamente por ter um caráter neutro, pode ser acessado por qualquer dispositivo e é extremamente leve ao ser comparado com outros formatos (SOAP, por exemplo).

Falando em neutralidade, independência de plataforma, o Giuliano continua seu minicurso sobre desenvolvimento para Android com o Rad Studio. Nesta terceira parte é feita uma introdução ao framework PhoneGap, que permite a geração de aplicativos mobile para as mais diversas plataformas utilizando um mesmo código fonte em HTML 5.

Para encerrar o Guinther nos presenteia com algo muito importante no ciclo de vida de um projeto, o gerenciamento de risco. Muitas vezes não consideramos certos fatores e simplesmente "saímos programando". Veja neste artigo as boas práticas envolvidas para ter o domínio sobre seu projeto. Boa leitura!

Paulo Quicoli
pauloquicoli@gmail.com
Editor Geral
quicoli.wordpress.com
twitter.com/pauloquicoli

Fale com o Editor!

É muito importante para a equipe saber o que você está achando da revista: que tipo de artigo você gostaria de ler, que artigo você mais gostou e qual artigo você menos gostou. Fique a vontade para entrar em contato com os editores e dar a sua sugestão!

Se você estiver interessado em publicar um artigo na revista ou no site ClubeDelphi, entre em contato com os editores, informando o título e mini-resumo do tema que você gostaria de publicar:

Paulo Quicoli - Editor da Revista
pauloquicoli@gmail.com

MVP

Assine agora e tenha acesso a todo o conteúdo da DevMedia: **www.devmedia.com.br/mvp**

DevMedia

NOTA DE RETRATAÇÃO

Ao Sr. Luciano Nunes

A Neofício Editora Ltda - ME, responsável pela edição da revista ClubeDelphi, **na pessoa de seu sócio, Sr. Gladstone Matos Gomes**, por força da determinação judicial referente ao Processo nº 0126779-85.2006.8.19.0001 da 16ª Vara Cível da Comarca da Capital/RJ, movido pelo Sr. Luciano Nunes em face do Sr. Gladstone Matos Gomes, **torna de conhecimento público a seguinte RETRATAÇÃO nos termos que seguem**.

Foi publicado na revista ClubeDelphi, edição 54, ano IV, um COMUNICADO dizendo que terceiros teriam obtido informações do banco de dados do ClubeDelphi, o que seria objeto de ação judicial e indicando-se o site www.clubedelphi.net, onde os interessados teriam acesso ao "AVISO DE RESPONSABILIDADE PÚBLICA".

O Sr. Luciano Nunes entendendo ter sido vítima de ofensas caluniosas, ingressou com o processo acima referido, em face do Sr. Gladstone Matos Gomes, sócio da Neofício Editora Ltda - ME e responsável pela edição e pelo conteúdo das revistas ClubeDelphi.

Apurados os fatos em referido processo, o Excelentíssimo Senhor Doutor Juiz de Direito da 16ª Vara Cível da Comarca da Capital/RJ, bem como os Respeitáveis Desembargadores do Tribunal de Justiça do Estado do Rio de Janeiro entenderam que os fatos veiculados ao público não são verdadeiros e ofenderam a honra do Sr. Luciano Nunes. Ainda, consignou-se que por intermédio de outro processo, (nº 2004.001.095541-7 com trâmite perante a 1º Vara Empresarial da Capital), afastou-se a responsabilidade do Sr. Luciano Nunes na subtração de dados do ClubeDelphi.

Assim, o Sr. Gladstone Matos Gomes, por meio da presente publicação se retrata perante o Sr. Luciano Nunes, tornando pública a inexistência de vinculação e responsabilidade do mesmo pelos fatos ocorridos, bem como dando cumprimento a ordem judicial emanada nos autos do processo nº 0126779-85.2006.8.19.0001 da 16ª Vara Cível da Comarca da Capital/RJ.

Da mesma forma, nos termos de referida decisão concede ao Sr. Luciano Nunes o direito de resposta a presente retratação, com manifestação em até 15 dias desta publicação.

Atenciosamente,

Gladstone Matos Gomes
Neofício Editora

Gerência de configuração

Versionamento com Delphi XE 2

Resumo DevMan

De que se trata o artigo:

Este artigo trata do entendimento da gerência de configuração e especialmente do controle de versão que é a prática principal envolvida. A integração do Delphi XE 2 com o gerenciador de versão Subversion é abordada e mostrada na prática.

Em que situação o tema é útil:

Este tema é útil para a boa administração de um projeto, onde se tem o controle do que é alterado, por quem é alterado e qual o motivo, além de facilitar o trabalho em equipe gerenciando o código fonte que é manipulado de forma simultânea.

Gerência de configuração:

Quando se trabalha em equipe, a divisão de tarefas é um trabalho comum. Essas tarefas exigem que código fonte seja alterado, consequentemente é possível que para realizar a tarefa X e Y um mesmo código precise ser modificado. O que fazer quando dois desenvolvedores precisam desse mesmo código? O que fazer quando é necessário ter duas versões distintas de um mesmo projeto sendo implementadas? Este artigo trata da gerência de configuração, um esclarecimento do que é tratado nela e como o Delphi XE 2 oferece suporte a uma de suas áreas, o controle de versão.

Muitos resumem gerência de configuração a apenas controle de versão de código. Contudo essa matéria é muito mais abrangente. Ela envolve os elementos empregados para o desenvolvimento de um software, como documentação, requisitos e o próprio código fonte. Assim como um software sofre alterações, todos os outros elementos também sofrem, essas mudanças precisam ser gerenciadas para que a equipe envolvida tenha controle sobre o que é desenvolvido e assim consiga manter o projeto estável e em evolução.

A gerência de configuração de software também é conhecida como GCS e é composta pelas seguintes atividades que poderão ser identificadas no decorrer do artigo:

- Estabelecer uma ligação entre os elementos de desenvolvimento que podem sofrer alteração;
- Definir um processo/mecanismo para controlar as versões dos elementos de desenvolvimento;
- Permitir que auditorias possam ser realizadas, para investigar o que foi modificado;
- Garantir que o processo criado é seguido por todos;
- Tornar pública aos envolvidos, as informações de um projeto.

Agora temos uma visão mais abrangente sobre a GCS, podemos dizer então que a mesma é uma atividade que pode garantir qualidade ao software durante seu desenvolvimento, isto porque através de sua aplicação controlamos de forma sistemática as mudanças solicitadas, mantendo integridade e rastreabilidade.

Composição

As atividades da GCS podem ser agrupadas em tipos: controle de versão, controle de mudanças e integração contínua.

Integração contínua: é um processo que tem por objetivo a geração automática do executável do software assim que mudanças solicitadas sejam implementadas. Essa geração envolve a execução de testes automatizados e até mesmo a construção do pacote de instalação do software. Existem ferramentas que fazem esse trabalho, elas podem ser configuradas para acessar o sistema gerenciador de versão em uso e a cada operação de check-in (confirmação de alteração de código) ou período de

tempo, recuperar a versão mais atual do código fonte, compilá-lo e processar o que é necessário. Para Delphi temos o FinalBuilder.

Controle de mudanças: É um serviço auxiliar e complementar ao controle de versão. Nele são administrados os requisitos, fornecendo meios para identificá-los, rastreá-los, analisar onde a mudança de requisito vai impactar, aceitar ou rejeitar esses requisitos. Existem no mercado ferramentas especializadas, como Microsoft Team Foundation Server, Trac, Bugzilla, Caliber e outras.

Controle de versão: É o pilar de sustentação de toda a gerência de configuração, apoiando as atividades de controle de mudança e integração contínua. Nele é que identificamos, armazenamos e gerenciamos os itens de configuração e de suas versões por todo o ciclo de vida do software (concepção, desenvolvimento, manutenção). Um item de configuração nada mais é algo que envolve o desenvolvimento, pode ser a descrição de um requisito, um código fonte, um banco de dados. Ele deve manter um histórico de todas as alterações efetuas nos itens de configuração, deve permitir também a criação de labels (rótulos) e branches (ramificações) do projeto e ainda permitir que um determinado momento de uma configuração possa ser recuperado.

Controle de versão

Durante a fase de desenvolvimento de um software enfrentamos vários problemas, como o não conhecimento completo de uma ferramenta, falta de entendimento dos requisitos, requisitos incorretos, testes incompletos e também a falta de controle de versão. Sim, a falta desse controle é um problema.

Vamos ilustrar aqui algumas situações que podem acontecer. Imaginando que você tenha realizado uma série de ajustes no sistema, todos solicitados pelo seu cliente, mas esse, depois de ver todo o sistema alterado (com aquilo que ele pediu) resolve que não quer mais a alteração. Ou, uma série de recursos novos estão em implementação em seu sistema, porém esses ainda não estão finalizados, então seu cliente pede algo urgente. Como podemos lidar com essas situações? Restaurando uma cópia de segurança, comentando código não acabado? Mas quem irá garantir a estabilidade desse código? Essas são algumas situações que qualquer desenvolvedor que trabalhe sozinho pode enfrentar, mas tudo pode ficar mais complicado quando se tem um time de desenvolvedores. Nessa situação podemos ter código sendo sobrescrito acidentalmente, não é fácil identificar quais partes do código foram alteradas e quem realizou essas alterações, recuperar uma versão anterior é um processo não seguro, pois irá depender de cópias de segurança e até mesmo o planejamento de versões se complica. A solução para tudo isso está em um sistema de controle de versão.

Funcionamento geral

Todo sistema de controle de versão (VCS – Version Control System) mantém um histórico dos arquivos alterados. Além disso, permitem a comparação entre essas versões, mostrando o que foi modificado. Um VCS é considerado como uma ferramenta de colaboração, porque permite que várias pessoas acessem um mesmo recurso (arquivo), administrando as alterações realizadas nele. Assim, também oferece um controle de usuários.

Cada alteração feita é associada ao usuário que a fez, permitindo posteriormente identificar seus responsáveis, também é possível determinar os níveis de acesso e permissão desses usuários, isto é, podemos configurar quais arquivos um determinado usuário pode acessar e o que ele pode fazer (ler/escrever).

Existe um conceito de funcionalidade que cria um marco temporal nos arquivos, permitindo a qualquer momento restaurar o estado desses quando necessário. Cada VCS implementa esse conceito com um nome diferente, como label ou tag. Por exemplo, podemos criar um label indicando um momento na vida do projeto em que ele estava estável (sem bugs) e continuar fazendo as alterações. Se precisarmos da última versão estável basta recuperar os arquivos do ponto que foi marcado.

Outro conceito comum é o da ramificação (branch). Este permite criar versões de um mesmo projeto, dividindo-o em linhas de desenvolvimento diferentes. Assim, é possível ter, por exemplo, a versão 1 estável e sendo desenvolvida e uma versão 2 em andamento contendo novas implementações. Ainda é possível mesclar alterações de código entre essas ramificações. Além dessas ramificações, existe também a linha principal de desenvolvimento, de onde essas ramificações partem, conhecida como tronco(trunk).

O local onde as informações de versionamento, histórico de cada item que é versionado é chamado de repositório. Quando utilizamos um VCS não trabalhamos diretamente nesse repositório, mas sim em uma cópia local, comumente chamada de cópia de trabalho ou área de trabalho. Essa cópia é individual, cada desenvolvedor envolvido a possui. A **Figura 1** mostra a comunicação realizada entre cópia de trabalho e repositório.

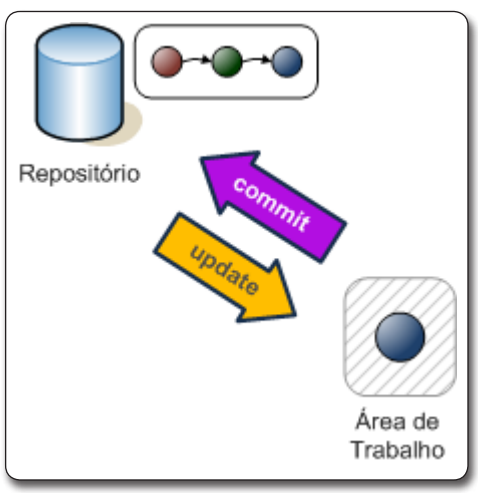


Figura 1. Comunicação com VCS [Fonte: http://www.pronus.eng.br]

A comunicação é feita por dois processos distintos. Um que envia uma nova versão e um que recupera uma versão. Essas operações são conhecidas como commit (envio) e update (recuperação) ou respectivamente como check-in e check-out.

Nota

Sempre que uma nova versão de arquivo é enviada ao repositório através de uma operação de check-in, a versão anterior não é eliminada. Ela fica armazenada permitindo ao usuário restaurá-la a qualquer momento, através de uma operação de check-out. Essa versão é chamada de revisão.

Um processo de configuração

Mais do que ter ferramentas para gerenciamento de configuração, é preciso deixar bem claro para a equipe de desenvolvimento qual é o processo de gerência em si. Isso informa a todos como proceder quando for necessário planejar uma nova versão e quando for necessário corrigir um problema. A **Figura 2** demonstra a sugestão de um processo.

Temos um trunk principal que está na versão 10. Nessa abordagem ele contém a versão homologada, que está em uso nos clientes. A equipe ao receber um conjunto de requisitos que sejam correções cria então um novo branch, chamando-o de versão 10.1, a partir do trunk. Isso garante e integridade do código fonte da versão que está em uso, pois as correções são realizadas no branch e não no trunk.

Enquanto isso outros desenvolvedores podem programar novas funcionalidades para a versão seguinte. Para isso eles também realizam um branch, de nome versão 11. Quando o trabalho de correção do branch 10.1 finalizar, suas alterações devem ser subidas ao trunk (esse é o jargão utilizado). Uma vez subidas, é criado um label no trunk, um marco, dizendo que agora ele é a versão 10.1 e para que a nova versão em desenvolvimento também as receba, é preciso baixar essas correções do branch 11. Quando a versão 11 for encerrada, a mesma também é subida ao trunk. Este é um trâmite bem comum e não é difícil de implementar em sua equipe de trabalho.

Subversion

Os sistemas gerenciadores de versão trabalham no formato cliente/servidor, onde figuram como servidores e para acessá-los é necessário um cliente. O Delphi XE 2 implementa um cliente de acesso ao sistema Subversion. Este é free e open source, sendo originalmente criado e mantido pela CollabNet. A versão distribuída pela CollabNet é um pouco complexa de ser instalada e para facilitar a adesão ao uso do Subversion, outras empresas criaram instaladores e até mesmo servidores SVN especializados. Foi o caso da VisualSVN. Essa empresa disponibiliza em seu site o download de um aplicativo servidor SVN free destinado a máquinas que utilizam o sistema operacional Windows. Vamos utilizar essa versão neste artigo. Após o devido download (ver seção **Links**), basta executar o instalador. Durante a instalação podemos configurar (**Figura 3**) a pasta que irá conter os repositórios que podem ser criados e a forma de autenticação ao servidor. Deixamos tudo com as configurações padrões.

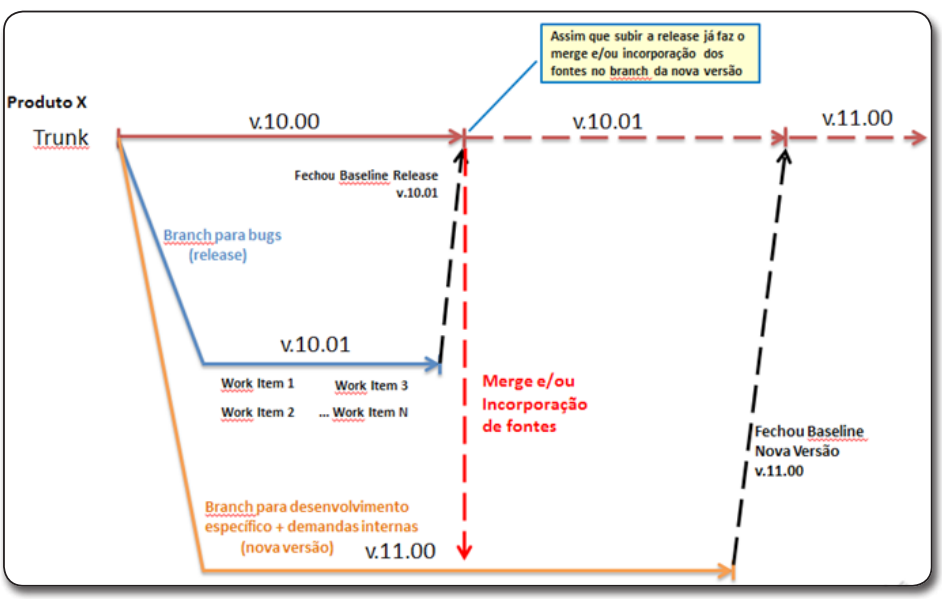


Figura 2. Processo de gerência de configuração

Nota

O VisualSVN durante sua instalação sugere um porta TCP de comunicação, a 443. Caso ela já esteja em uso por outro aplicativo, o próprio instalador irá requerer a mudança de porta.

O último passo do instalador é informar se desejamos ou não, iniciar a execução do servidor. Ao escolhermos sim o mesmo é iniciado (**Figura 4**).

É possível visualizar a situação atual, que mostra o caminho de acesso do servidor, a quantidade de usuários já cadastrados, opções de log e a quantidade de repositórios existentes.

Para continuar com a configuração do servidor vamos definir um usuário, bastando clicar na opção Create new user. A tela que surge, informamos um login e senha e confirmamos a criação. Para criar um repositório, acessamos a opção Create new repository, esta exibe uma tela (**Figura 5**) para informamos o nome do repositório e se desejamos que seja criada a estrutura padrão de pastas do SVN.

Essa estrutura é compreendida pela pasta trunk, que no processo sugerido pela **Figura 2** contém o código mais recente e funcional, a pasta branches onde serão criadas as ramificações para novas versões e tags, que irá conter versões específicas. Como pode ser visto, foi definido o nome de ProjetoTeste ao repositório e foi marcada a opção para criação padrão de pastas.

Configurando o Delphi XE 2

O suporte ao Subversion é implementado por um projeto open source chamado Rad Studio Version Insight. Ele é free e open source, sendo atualizado constantemente. Para exemplificar, a versão do SVN que foi instalado é a 1.7, que originalmente não é

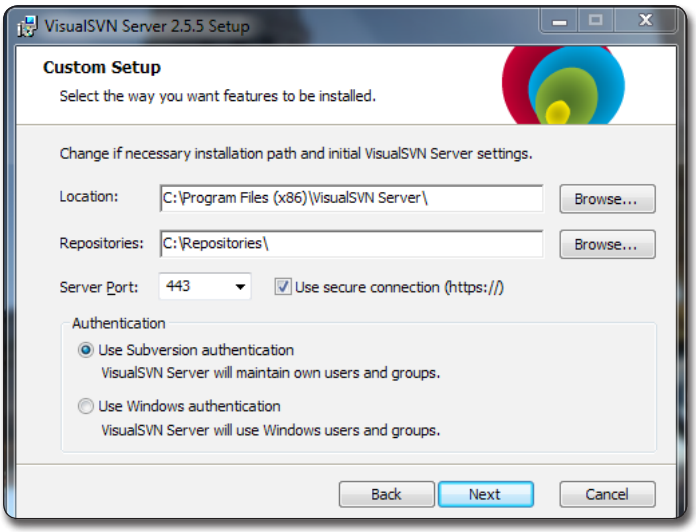


Figura 3. Instalando o VisualSVN

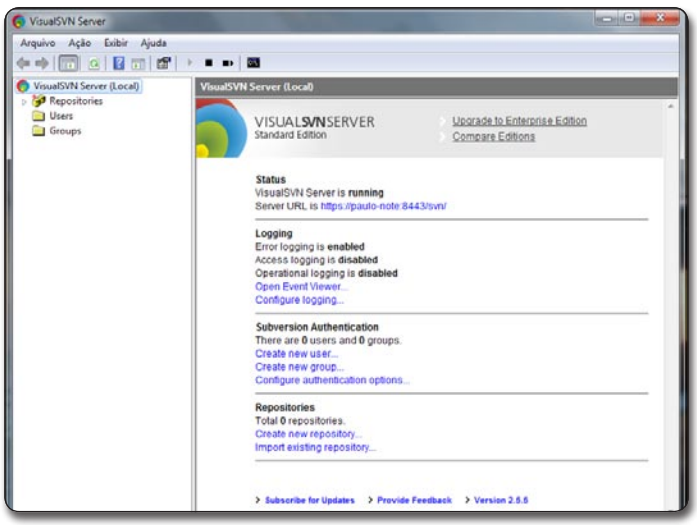


Figura 4. Servidor SVN em execução

suportada eficientemente pelo Delphi XE 2, é preciso atualizá-lo. No site do projeto temos o link para atualização (veja seção **Links**), que no momento da edição deste artigo está na versão Beta 9.1. Após seu download, sua instalação é simples, consistindo de um aplicativo que deve ser executado com o Delphi XE 2 fechado. Esse aplicativo atualiza as versões XE e XE 2 do Delphi. É importante frisar que essa atualização traz a possibilidade de utilizar outros controladores de versão, como Git e Mercurial **(Figura 6)**. Como estamos utilizando o Subversion, desmarcamos a opção Git e Mercurial.

Explorando o versionamento

É necessário termos uma pasta já configurada como cópia de trabalho, para que possamos manter nela o fonte de um projeto. No meu File>Open From Subversion (checkout) do Delphi, podemos criar essa cópia. Na tela que surge informamos o caminho da pasta trunk do repositório. Isso é obtido na tela do VisualSVN Server, ao abrir a pasta Repositories>ProjetoTeste e na sequência a pasta trunk (**Figura 7**).

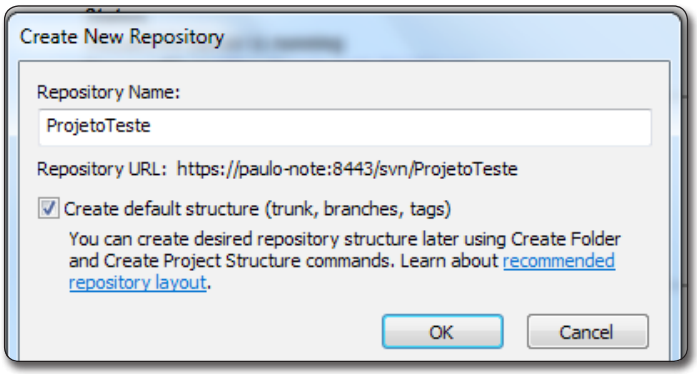


Figura 5. Criando um repositório

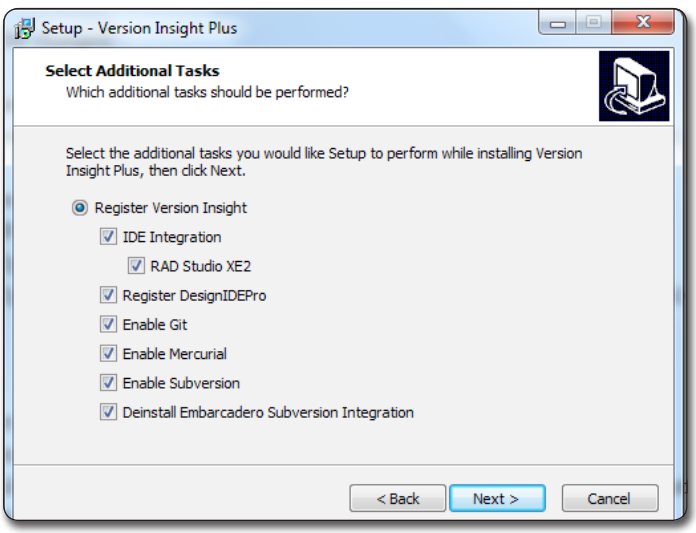


Figura 6. Atualizando o suporte ao SubVersion

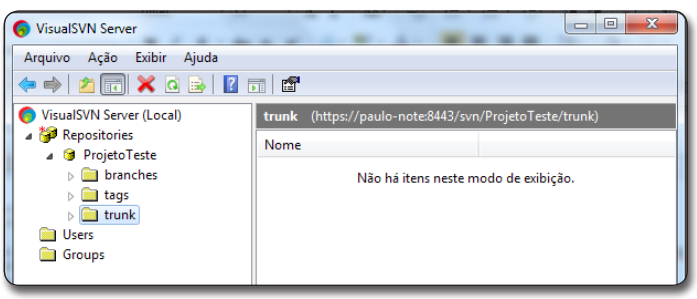


Figura 7. Caminho da pasta trunk do ProjetoTeste

Informamos esse caminho e uma pasta local (**Figura 8**) que receberá o checkout. O checkout atualiza uma pasta que é cópia de trabalho com os fontes que estão no repositório informado e cria um vínculo entre essas suas pastas (repositório e cópia local).

Criamos então um novo projeto do tipo VCL Forms Application. Seu formulário principal foi configurado como mostra a **Figura 9**. Temos simplesmente um botão que exibe uma mensagem, como é mostrado a seguir:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage('Fui inserido na versão 1');
end;
```

Ao salvar o projeto o salvamos na pasta que configuramos anteriormente. Vamos utilizar esse projeto para explorar situações diárias no controle de versão. A primeira delas é incluir o código fonte no repositório escolhido. Clicamos com o botão direito do mouse sobre o mesmo e escolher o item Subversion Import (**Figura 10**).

É possível notar na guia Import que somente os arquivos que realmente precisam ser versionados são sugeridos no commit (ou check-in). A integração é inteligente o suficiente para ignorar arquivos temporários e .dcu (arquivos compilados). No campo URL of repository já temos a o endereço da pasta trunk do projeto, podemos incluir uma mensagem informando do que se trata esse primeiro commit (envio de atualizações). No campo Import Comment vamos informar "inclusão do projeto no controle de versão" e então clicar em Import. Será solicitada a senha do usuário anteriormente criado no VisualSVN, informe e confirme. Os fontes serão enviados ao Subversion.

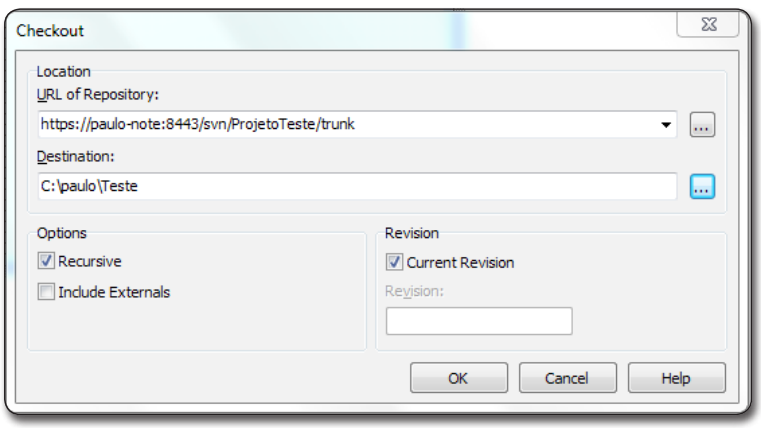


Figura 8. Realizando um checkout

Figura 9. Formulário principal

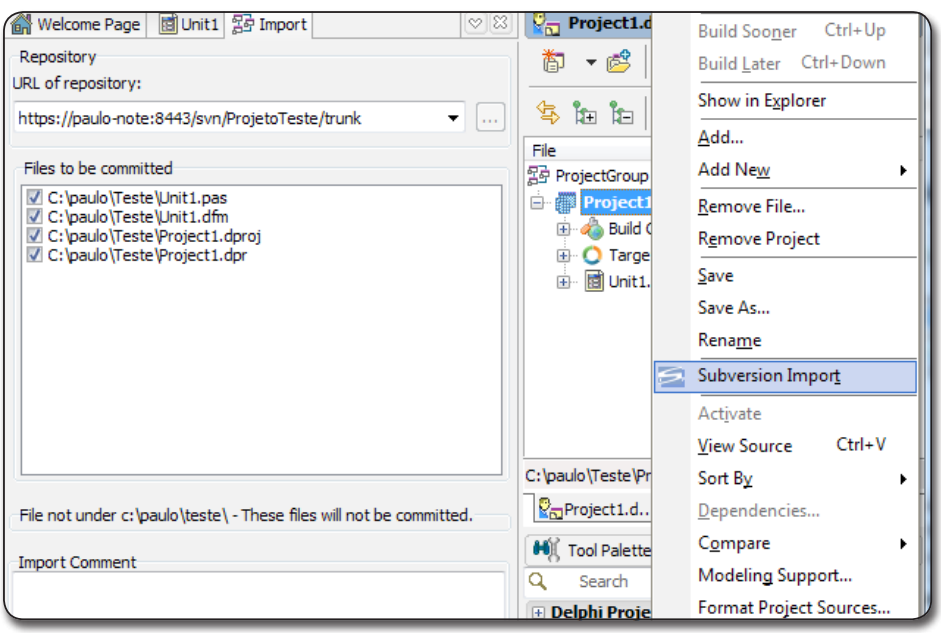


Figura 10. Opção Subversion Import

Com o código fonte versionado, algumas coisas mudaram no IDE do Delphi, veja a **Figura 11**.

Temos uma nova guia chamada Subversion, ela exibe as mensagens que o servidor retorna, indicando se as operações de controle de versão foram bem sucedidas. E por todo IDE temos o indicativo da situação dos arquivos em relação ao controle de versão. O marcador verde indica que o código marcado está sincronizado com o repositório, ou seja, temos a mesma versão. Se alterarmos algo no código fonte, ou seja, se modificarmos a versão, esse marcador também refletirá esse estado. Vamos fazer uma simples alteração para visualizar esse aviso. No clique do botão vamos incluir um novo código, como é mostrado a seguir:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage('Fui inserido na versão 1');
  ShowMessage('Sou uma alteração');
end;
```

Ao salvarmos esse código, o símbolo de status se transforma em um sinal de exclamação vermelho com o texto Modified. Para enviar essa atualização ao repositório, clicamos com o botão direito no projeto (via Project Manager), escolhemos o menu Subversion>Commit>Files in this Project. Essa opção irá detectar todos os arquivos alterados e os mostrará em uma janela específica (**Figura 12**).

Nessa janela podemos selecionar quais arquivos queremos enviar ao repositório e adicionar um comentário indicando do que se trata a modificação feita. É nesse ponto que encontramos o item rastreabilidade da gerência de configuração. Começamos a saber o que levou, qual requisito/necessidade, a essa alteração de código fonte. Ao clicarmos no botão Commit a nova versão é enviada.

Visualizar o histórico de alterações é uma operação comum quando se precisa saber o que foi alterado, quem alterou e a causa disso em um código fonte. A integração do IDE permite um rápido acesso a essas informações.

No rodapé do arquivo Unit1.pas temos a guia History que exibe uma janela especial, como mostra a **Figura 13.**

As informações importantes no grid apresentado são Revision (revisão), Date (data), Author (autor) e Comment (comentário). Identificamos cada versão existente para o arquivo selecionado, essas são as revisões. Cada novo commit gera uma nova versão que é tratada como revisão. Vemos quem realizou o commit e a mensagem informada, como mostra a revisão 2 indicando que esse foi o momento que subimos o projeto.

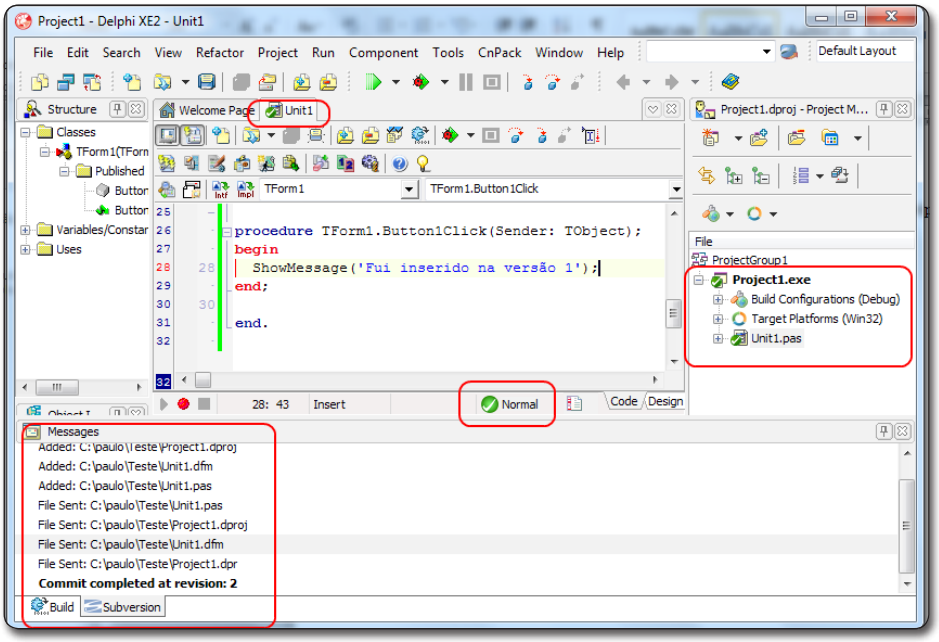


Figura 11. Destaques do Subversion

Junto com o histórico do repositório temos o histórico que é criado pelo próprio Delphi. Esse não tem nada a ver com o histórico do repositório, é criado pelo Delphi cada vez que o arquivo for salvo. Este histórico criado pela IDE fica dentro da pasta oculta chamada __history no mesmo local onde o projeto está salvo.

Nessa mesma tela também é possível visualizar a diferença de conteúdo entre as versões. Basta acessar a guia Differences (**Figura 14**).

Nota do DevMan

Cada vez que um arquivo é salvo dentro do IDE do Delphi, um cópia do arquivo é criada dentro da pasta __history. Esta pasta é criada automaticamente pelo Delphi no local onde o projeto está salvo. Este recurso não é muito interessante quando o projeto já está sob o controle de um sistema controlador de versão. Para desativá-la, é preciso acessar Tools + Options do Delphi e dentro da opção Editor options desmarcar o item Create backup files.

Nessa janela temos duas partes onde escolhemos o que comparar. Como é mostrado escolhemos em Differences From a revisão 2 e em To, a revisão 3. Logo a baixo é exibido que existe de diferente, ou seja, da revisão 2 para a 3 temos a linha 29 destacada em verde, com o um sinal de "+" indicando que a mesma foi adicionada.

Para simularmos uma operação de dois usuários alterando um mesmo código fonte, vamos criar outra cópia de trabalho e recuperar nela os fontes do repositório.

No Delphi fechamos o projeto atual (File>Close All). E assim como foi feito anteriormente, acessamos o menu File>Open From Subversion. Informamos o caminho do repositório existente e o caminho local, que agora deve ser diferente. Ao confirmarmos o IDE acessa o repositório e recupera o código fonte existente. Nesse momento o Delphi nos permite escolher o projeto a abrir. Como

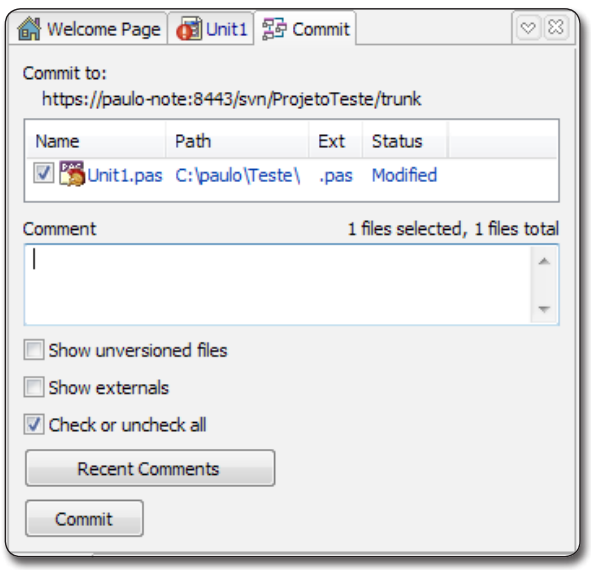


Figura 12. Efetuando um commit

no repositório só temos um projeto, somente esse é que aparece. Confirmamos e vemos então outra janela que mostra tudo o que foi trazido do repositório. Os arquivos estão sinalizados com o Check verde, indicando que estão em sincronia. Vamos incluir novamente no click no botão outra mensagem e então salvar o código, como é mostrado a seguir:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage('Fui incluído fora de ordem');
  ShowMessage('Fui inserido na versão 1');
  ShowMessage('Removi o texto anterior');
end;
```

Desta vez incluímos um texto no início e modificamos uma mensagem anterior. Mais uma vez nos é sinalizado que o arquivo foi modificado (exclamação em vermelho), então vamos enviar essa modificação ao repositório. Clicamos com o botão direito no projeto e escolhemos Subversion>Commit>Files in this Project. Na tela de commit vamos adicionar um novo comentário: "Alterações simuladas" e finalizar a operação com o botão Commit.

Para realizarmos uma operação de update, que é a ação de atualizar a cópia local para que os arquivos reflitam a última versão existente no repositório, vamos abrir novamente o projeto da primeira pasta que criamos. Na verdade o Update faz um pouco mais do que simplesmente atualizar para a última versão, pois no caso de existir alterações na cópia local, elas não serão perdidas, ao invés disto suas alterações serão mescladas com a última versão existente.

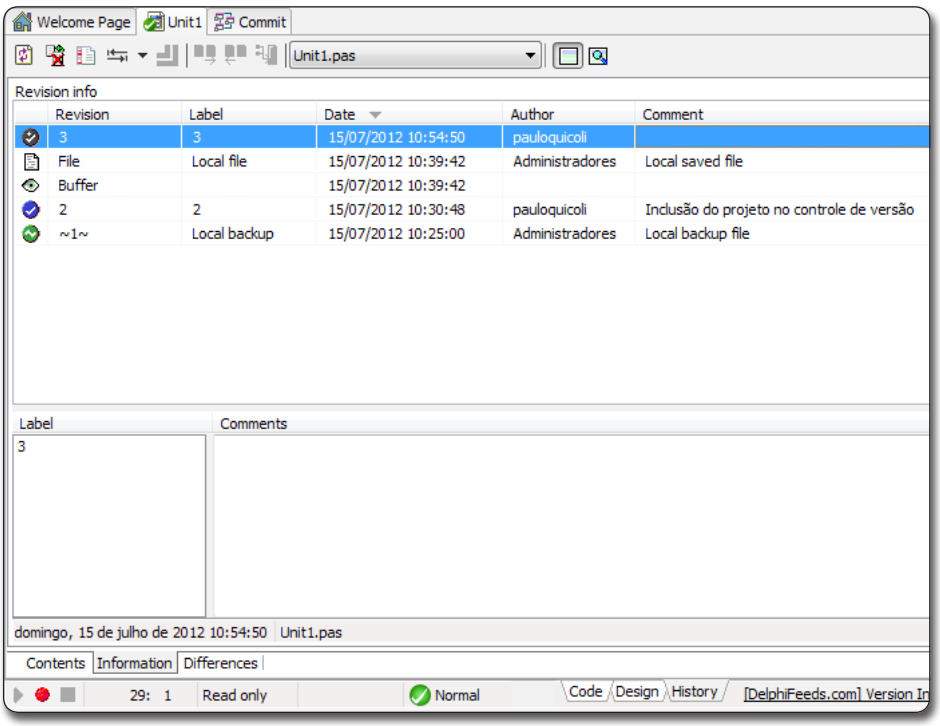


Figura 13. Histórico de versão

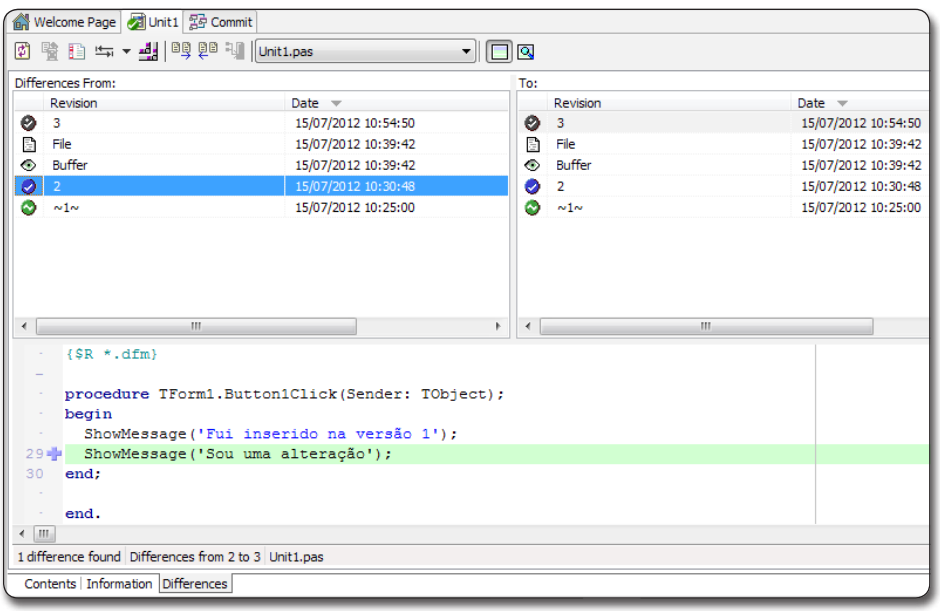


Figura 14. Comparando versões

Ao abrirmos o projeto clicamos com o botão direito no mesmo e escolhemos a opção Subversion>Update>Files in this Project. Durante o processo de atualização é exibida uma janela que mostra o que é feito nesse momento (**Figura 15**).

Podemos ver que temos uma revisão 4 e ela foi recuperada para nossa cópia local que estava com a revisão 3. Ao completar temos todo o código fonte ajustado.

Existe um recurso na integração que permite visualizarmos a origem de cada alteração no código de fonte, é a opção Show Blame exibida na **Figura 16**.

Ela exibe uma coluna no canto esquerdo do código fonte mostrando em destaque as linhas alteradas. Vemos o número de revisão em que essa alteração ocorreu, a data, o usuário e a mensagem. Isso acaba com aquelas situações onde você tem seu código alterado por alguém do time mas não se sabe o porquê.

Mas tudo são flores no gerenciamento de versão? Não. Podem existir conflitos que a ferramenta não consiga resolver de forma automática. O Subversion é um sistema controlador de versão que trabalha com mesclagem otimista. Esta filosofia de mesclagem acredita que os conflitos de edições serão tão raros que a mesclagem pode ser feita de forma automática. A primeira impressão é que esta filosofia não funciona, mas na prática é assim que os projetos vêm sendo gerenciados com o Subversion desde o início.

Esses conflitos podem ser separados em duas categorias: conflitos de arquivos e conflitos de linhas dos arquivos. Os conflitos de arquivos são comuns e acontecem quando dois programadores alteram o mesmo arquivo, porém em linhas diferentes. Neste caso o Subversion será capaz de mesclar as alterações automaticamente. Quando dois programadores fazem alterações na mesma linha do mesmo arquivo, o Subversion não será capaz de fazer a mesclagem sozinho e você deverá dizer para o Subversion qual das duas edições será descartada.

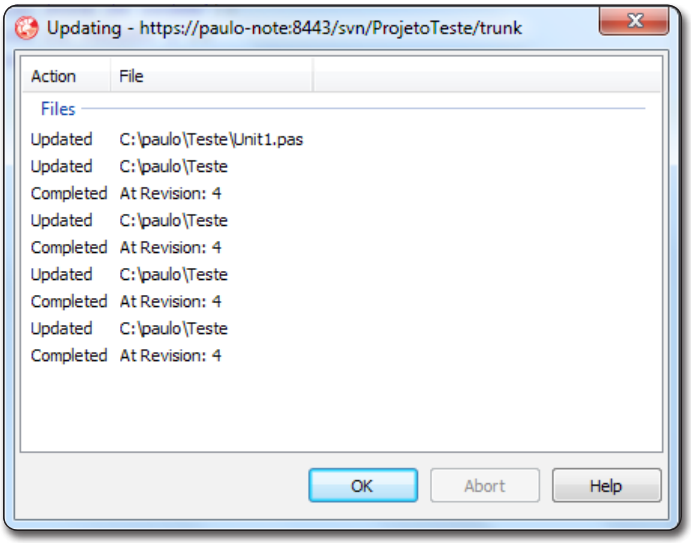


Figura 15. Tela de atualização

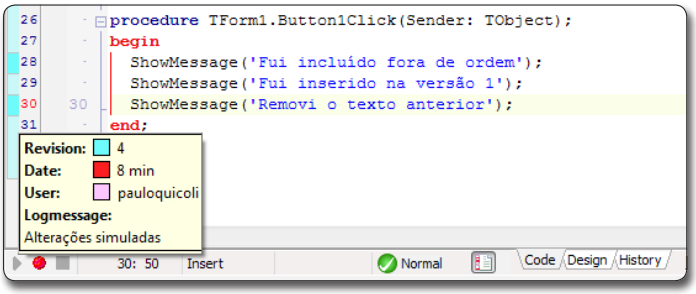


Figura 16. Opção Show Blame

Conclusão

Neste artigo vimos uma introdução à gerência de configuração, focando apenas o controle de versão e como o Delphi XE 2 torna sua aplicação rápida. Temos outros assuntos pertinentes, como o controle de requisitos e a integração contínua onde cada um por si só dariam vários artigos. Até mais!

Paulo Quicoli
pauloquicoli@gmail.com
Editor da revista ClubeDelphi e editor técnico .Net Magazine. Formado em processamento de dados pela FATEC-TQ. Atua como analista programador de sistemas e coordenador técnico da equipe de desenvolvimento .NET da Siplan Control-M (www.siplancontrolm.com.br) unidade Jaboticabal, e professor da FATEC, campus Taquaritinga. Blog: http://quicoli.worpress.com

VisualSVN
http://www.visualsvn.com/server/download/

Rad Studio Version Insight
http://www.bitcommander.de/blog/index.php/category/delphisvn/

FinalBuilder
http://www.finalbuilder.com

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Faça um *upgrade* em sua carreira

Em um mercado cada vez mais focado em qualidade, ter conhecimentos aprofundados sobre requisitos, metodologia, análises, testes, entre ouutros, pode ser a diferença entre conquistar ou não uma boa posição profissional. Sabendo disso a DevMedia lançou uma publicação totalmente especializada em Engenharia de Software. Todos os meses você pode encontrar artigos sobre Metodologias Ágeis; Metodologias tradicionais (document driven); ALM (application lifecycle); SOA (aplicações orientadas a serviçoes); Análise de sistemas; Modelagem; Métricas; Orientação à Objetos; UML; testes e muito mais. **Assine Já!**

DevMedia

Visual Metro no Delphi

Conheça o pacote TMS Metro Controls

Atualmente, os mais diversos softwares tem atingido um padrão de maturidade onde fazer o que precisa ser feito não é mais um diferencial. Isto quer dizer que, se seu sistema faz um controle de estoque, existem outros que também fazem. Ao tentar fazer uma venda outros fatores devem entrar no jogo, como suporte técnico e valor a ser cobrado. Existem outros dois fatores que também podem determinar uma venda: usabilidade e aparência.

Usabilidade

A usabilidade também é conhecida como experiência de uso, experiência do usuário ou simplesmente UX. Ela trata da interação entre um software e uma pessoa. É impossível falar de aparência ou usabilidade sem citar o próprio Windows. Desde sua criação seu visual vem passando pelas mais diversas inovações, todas, buscando a melhor usabilidade. Assim, é possível ver que usabilidade e visual estão relacionados.

O Delphi utiliza a VCL, sua biblioteca de componentes visuais, para criar sua interface. A VCL nada mais é do que a utilização das APIs visuais do Windows, assim, seu comportamento padrão é um reflexo do visual do Windows.

Inovar a usabilidade de uma parte de um sistema pode trazer consequências inesperadas. Vamos tomar por reflexão a evolução do sistema operacional Windows, começando com a primeira versão lançada **(Figura 1**) em 1985. Os computadores tinham um papel um pouco diferente do que têm hoje na vida das pessoas, eram utilizados mais em empresas e localmente, não existia a internet da forma como a conhecemos. O Windows era um software que criava uma utilização visual sobre os recursos existentes do DOS, que era o sistema operacional na época. Se você nunca viu o DOS, basta abrir o prompt de comando em seu Windows e terá uma ideia.

Sua usabilidade era voltada ao uso de teclado apenas, o mouse era opcional e não muito popular. Na verdade os usuários experientes da época o achavam ineficiente, meio atrapalhado e difícil de aprender a usar. Muitas revistas especializadas fizeram críticas duras como: "O mouse é uma boa ideia, mas seu valor para usuários de negócio é dúbio" – Revista PC Week, "Não existem evidências que as pessoas queiram usar estas coisas

Resumo DevMan

De que se trata o artigo:

Neste artigo abordamos a evolução visual das aplicações, em especial do sistema operacional Windows e a aplicação da linguagem visual Metro utilizando os controles visuais da TMS.

Em que situação o tema é útil:

Em melhorar a usabilidade dos sistemas, manter sua interface moderna e agradável aos usuários, buscando conformidade com conceitos atuais de apresentação de informação, podendo se tornar um diferencial em uma disputa comercial.

Visual Metro no Delphi:

O Delphi construiu seus pilares sobre o desenvolvimento RAD de aplicações desktop para Windows. Seus controles visuais (VCL) são implementações de controles nativos Windows, assim, seguem sua aparência padrão. Mas as interfaces gráficas estão mudando, o próprio Windows está mudando. Temos agora um novo padrão visual chamado Metro que explora outros modos de interação e exibição de informação. Enquanto a Embarcadero não disponibiliza uma versão com controles nativos a essa interface, vamos conhecer o pacote TMS Metro Controls. Esses são controles que possuem um visual inspirado no novo modelo, fazendo com que aplicações tradicionais ganhem um novo apelo visual. Vamos desenvolver um pequeno aplicativo que irá explorar esses controles.

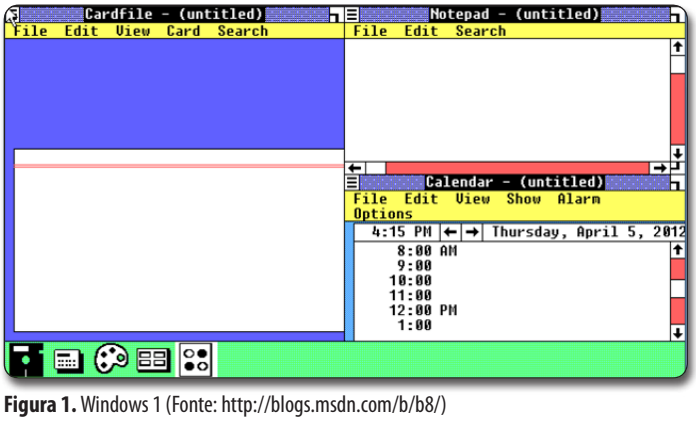


Figura 1. Windows 1 (Fonte: http://blogs.msdn.com/b/b8/)

(mouses)" – San Francisco Examiner, "O mouse faz um computador mais acessível, mais amigável, para certo tipo de público como Executivos? A resposta é não." – Revista Computerworld. E hoje é impossível imaginarmos o uso de um sistema atual sem sua existência.

Estes são apenas alguns exemplos do que uma inovação pode causar no imaginário das pessoas.

Já a versão 3 e 3.1 do Windows foram lançadas em 1990 e foram um sucesso na época. Outra inovação foi apresentada, um aplicativo que centralizava a abertura de programas, sua organização e a troca de aplicativos em execução (**Figura 2**).

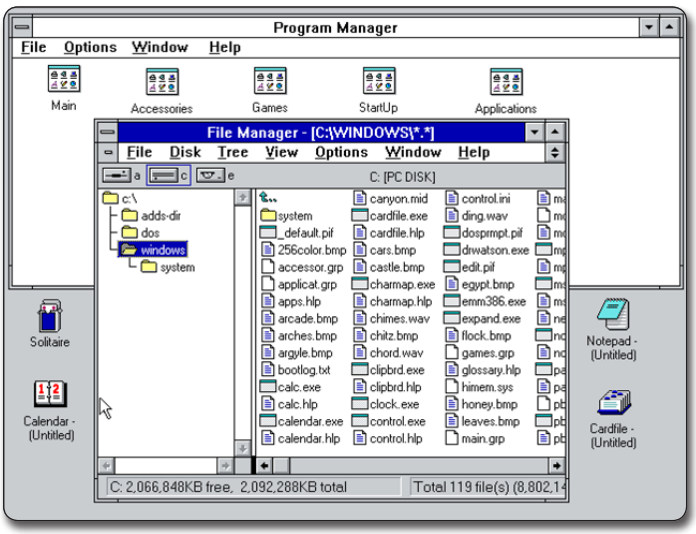


Figura 2. Windows 3 (Fonte: http://blogs.msdn.com/b/b8/)

A utilização de ícones no tamanho 32x32 foi outra novidade, mas causou atritos na mídia na época, justamente porque requeria o uso do mouse. Podemos perceber uma evolução na interatividade e na percepção do conteúdo. Arquivos que antes eram representados por apenas seu nome e uma extensão, agora possuem ícones que representam sua funcionalidade, melhorando a experiência de uso.

Nota

Em 1995 surge o Delphi 1, voltado para a plataforma Windows 16 bits. Foi a primeira ferramenta de desenvolvimento a ser intitulada como um ambiente RAD (desenvolvimento rápido de aplicações).

A versão 95 trouxe uma completa revolução visual e de usabilidade (**Figura 3**). Seus pilares se baseavam na introdução de novas interações representadas por objetos como um menu Iniciar, barra de tarefas, Explorer e área de trabalho. Sua importância é tão grande que esses conceitos ainda existem na versão 7 do Windows.

Não é preciso nem dizer que a inclusão de um botão Iniciar causou no início certa preocupação aos usuários, porque agora estariam presos a um único ponto de acesso a recursos do sistema operacional. O resultado disso é que o tempo consolidou a ideia proposta, tanto que nas versões posteriores até mesmo o termo "Iniciar" foi removido. Outro ponto que causou dificuldades inicialmente foi o menu de contexto, acessado pelo clique com botão direito do mouse sobre algum elemento. Esse conceito também era novo, incluindo o clique duplo que também não existia.

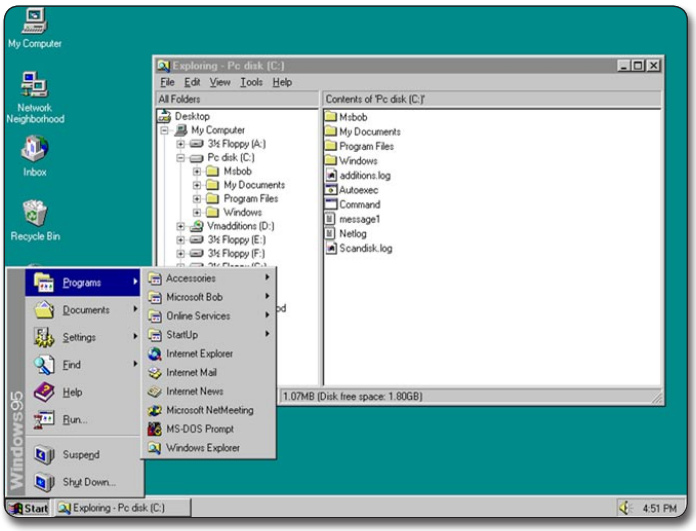


Figura 3. Windows 95 (Fonte: http://blogs.msdn.com/b/b8/)

Em 2001 o uso dos computadores estava passando por mudanças. As pessoas usavam seus computadores cada vez mais, navegavam na internet mais. Estavam mais conectadas do que durante o lançamento do Windows 95. Isso exigiu uma mudança na estratégia da Microsoft e assim surgiu o Windows XP, criando um novo menu Iniciar (**Figura 4**).

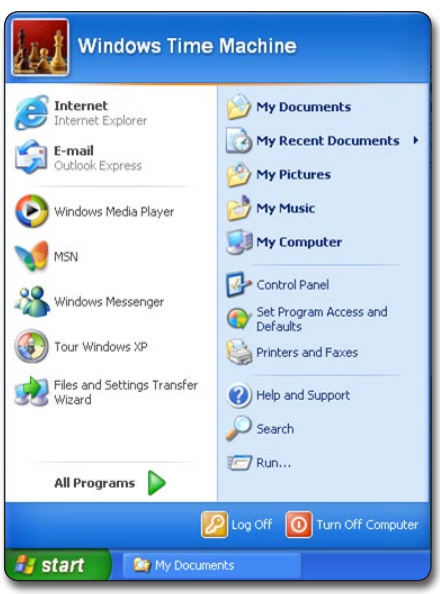


Figura 4. Menu Iniciar do Windows XP (Fonte: http://blogs.msdn.com/b/b8/)

Embora hoje o XP seja considerado um sucesso, na época vários desejaram voltar para uma versão anterior devido a essa mudança.

A versão seguinte, o Vista, foi lançada em 2006 e mudou novamente o visual, incluiu o estilo Aero, onde elementos realísticos são utilizados para refletir algo novo. O objetivo do Aero era tirar o foco visual das bordas das janelas e centrar no conteúdo da mesma, por isso as bordas parecem transparentes e vitrificadas (**Figura 5**).

Figura 5. Aero Glass (Fonte: http://blogs.msdn.com/b/b8/)

O Windows 7 trouxe mais recursos de usabilidade e visual. O mouse e teclado deixaram de ser a única forma de acesso ao sistema, o toque, ou melhor, o multi-toque foi adicionado. Outros detalhes também evoluíram, a barra de tarefas (**Figura 6**) permite uma experiência melhor na troca de aplicativos em execução e o menu Iniciar agora é usado para abrir os programas menos utilizados.

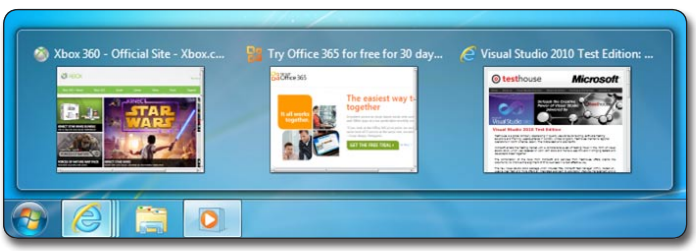


Figura 6. Windows 7 (Fonte: http://blogs.msdn.com/b/b8/)

O que vem por aí?

Não é novidade que o Windows 8 mudou completamente seu visual e usabilidade. Essa interface chama-se **Metro** e foi desenvolvida seguindo alguns princípios como consolidar tarefas comuns para acelerar o uso, pois, ninguém quer aprender novas formas de interagir com um conteúdo, excluir objetos gráficos que não tem real importância ao conteúdo, parecer o mais "natural" possível com algo digital, ou seja, não tentar imitar objetos como superfícies de alumínio ou madeira, utilizar animações mais naturais para mostrar que a interface é responsiva. O mais curioso disso tudo, é que na verdade essa linguagem visual foi criada a partir de coisas que já existem e são largamente utilizadas, como a sinalização de um metrô ou um aeroporto (**Figura 7**), onde o importante é fazer com que uma pessoa entenda a informação que está sendo exibida sem muito esforço, ou seja, informar com simplicidade e eficiência. Todo esse conceito também está relacionado com algo chamado NUI, Natural User Interface.

Figura 7. Sinalização inspiradora para o Metro

Natural User Interface

NUI é uma nova forma de pensarmos em como interagir com os sistemas computacionais. Nela são reaproveitadas as habilidades já conhecidas das pessoas para criar a interação com um conteúdo. Isso é alcançado criando interfaces que os usuários possam entender como metáforas, que expressam experiências do mundo real. Enquanto que o padrão atual de interface gráfica, chamado de GUI é voltado especificadamente para os dispositivos de entrada, a NUI foca no estilo de interação, onde qualquer tecnologia de interface pode ser utilizada (mouse, teclado, gestos, sons) desde que o foco seja reaproveitar habilidades.

O que é natural?

Em uma GUI tradicional temos botões, menus e ícones como elementos principais. Na NUI eles devem ocupar um papel secundário, deixando o usuário interagir diretamente com o conteúdo. É onde o natural entra. Uma interação natural é utilizar o que as pessoas já aprenderam no seu dia a dia. Um exemplo é o "tocar para ver se algo acontece". É muito comum vermos crianças aplicando isso no seu dia-a-dia, é uma habilidade inata de toda pessoa e que está muito relacionada à natureza curiosa do ser humano. Como isso pode ser utilizado em um sistema? Por exemplo, para visualizar os detalhes de alguma informação, onde essa informação é um conteúdo que irá reagir a essa interação. Muitas vezes uma habilidade inata não é o suficiente para a realização de uma tarefa, por exemplo, em manipulações de arquivos. Mesmo que o conceito de arquivo e pasta tenham suas raízes no mundo real, seu comportamento dentro de um

sistema não é o mesmo. Se esquecermos de salvar um arquivo seu conteúdo pode ser perdido. O risco do esquecimento pode ser reduzido ao utilizar mensagens de alerta ou ao implementar uma funcionalidade de auto salvamento, mas mesmo assim um usuário novato poderia se confundir. Através da NUI podemos melhorar essa experiência de uso.

Um projeto hipotético

Vamos imaginar um editor de textos que será utilizado em um tablet multi-toque, ele permitirá a criação de algum tipo de arquivo, salva o texto automaticamente a cada alteração, gera um nome para o arquivo e sabe onde salvá-lo.

Essas características já retiram do usuário novato a obrigação de conhecer os pormenores de uma tecnologia, já que sua obrigação será apenas a de escrever o documento.

Cada documento que o usuário criar será tratado como um conteúdo diferente, os elementos da interface serão os mais contextuais possíveis. Isto é, como a ideia do NUI é focar no conteúdo, esses controles visuais poderiam ficar ocultos e aparecer somente quando necessários.

Levando isso em consideração vamos imaginar como que um usuário poderia aumentar a fonte de um texto: Ele poderia arrastar seu dedo sobre o texto para aplicar a seleção. Isso geraria um efeito de destaque no mesmo e automaticamente um elemento de interação surgiria indicando como aumentar o tamanho da fonte utilizando o estilo de manipulação pinça (**Figura 8**). A mesma ideia pode ser utilizada para aplicar um zoom no documento em edição.

Figura 8. Manipulação estilo pinça

Outro recurso que deve ser utilizado para melhorar uma interação é a animação fluída, que representa os diferentes estados que um conteúdo pode ter. Por exemplo, ao navegar as páginas do documento um efeito de transição direcional (página locomovendo-se para trás ou para frente) pode ser aplicado ao avançar uma página e ao retroceder, dessa forma o usuário instintivamente irá tentar mudar de página arrastando seus dedos sobre a página. Estas são ideias, mas como aplicar esses conceitos em um aplicativo comercial?

Um projeto tradicional

Foi desenvolvido um programa simples, que consiste apenas de telas que mostram o padrão adotado pela maioria dos projetos desenvolvidos para Windows, utilizando Delphi. Na **Figura 9** temos a tela principal. Ela consiste de um menu principal que expõe as funcionalidades disponíveis e uma área central, onde geralmente os desenvolvedores adicionam o logo de suas empresas.

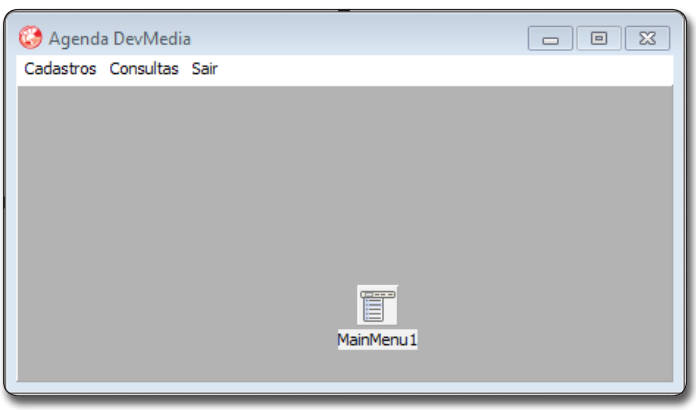


Figura 9. Menu principal tradicional

Quando um sistema possui muitas opções, também é uma prática comum adicionar uma ToolBar que contém as opções de uso mais frequentes, para facilitar o uso ao usuário. No nosso caso estamos simulando uma agenda que possui duas funcionalidades, cadastrar contatos e recuperar os aniversariantes. A tela de cadastro (**Figura 10**) segue o comportamento comum e aparência bem conhecida pelos desenvolvedores.

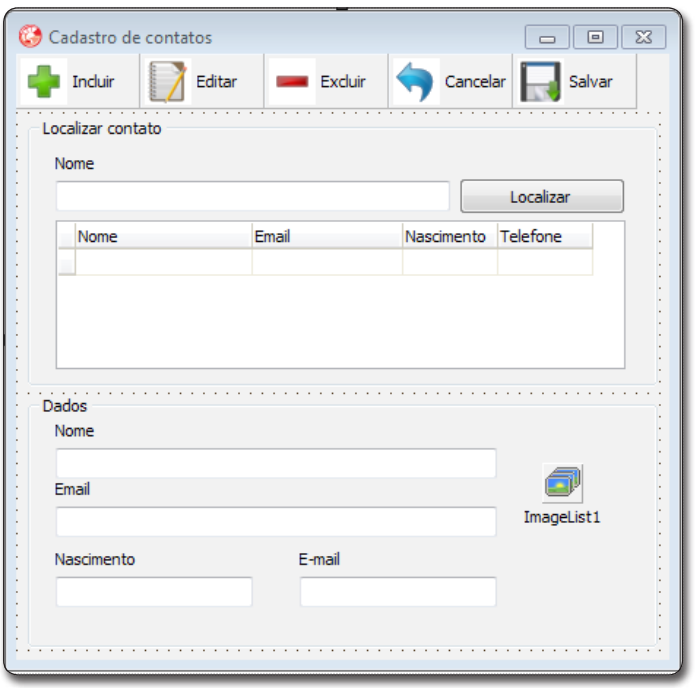


Figura 10. Cadastro de contatos

Temos agora o uso de ícones para representar ações. Nem todos criam uma barra personalizada para manutenção de dados, muito se dão por satisfeitos apenas com a barra de navegação padrão. A interação acontece da seguinte forma: o usuário pode localizar e selecionar um contato. Ao escolher alguém, ele pode editar seus dados pelo botão Editar ou remover por completo pelo botão Excluir. Para salvar um novo contato ou confirmar uma edição ele deve pressionar o botão Salvar. É uma prática muito comum emitir uma confirmação a respeito desse salvamento, informando que o mesmo ocorreu com sucesso. Também é uma prática comum solicitar uma confirmação ao excluir um registro. A busca de aniversariantes (**Figura 11**) também é bem similar. Através de um ComboBox o usuário escolhe o mês e o sistema recupera os devidos contatos.

A nova proposta

Vamos refazer o visual desse aplicativo utilizando alguns conceitos Metro, como uma janela minimalista, com poucas cores, enfatizando o conteúdo. Para isso não temos que começar do zero, a TMS desenvolveu um conjunto básico de componentes visuais que aceleram a adoção desse visual. Para começar é preciso realizar seu download (ver sessão **Links**) e instalar no Delphi através do setup fornecido. Sua instalação é simples, segue o padrão Windows (avançar, avançar, terminar).

Uma vez instalado é criado um novo tipo de projeto dentro do IDE. Ele pode ser acessado ao criar um novo projeto, na opção TMS Forms>TMS Metro Application (**Figura 12**). Foi isso que fizemos.

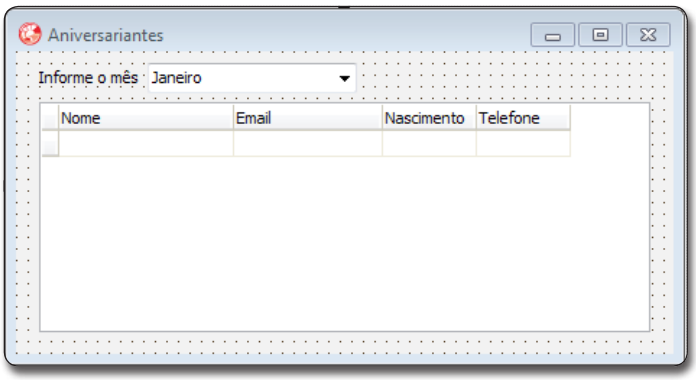


Figura 11. Busca de aniversariantes

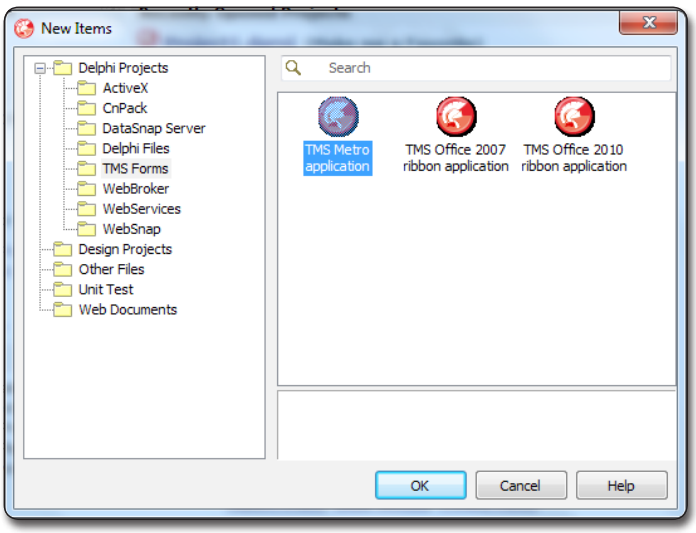


Figura 12. Novo tipo de projeto

Um novo projeto é criado e esse já possui um formulário principal (**Figura 13**). Ao executar a aplicação da maneira que é criada, já vemos a diferença desse formulário.

A primeira coisa que encontramos é que ele herda de TAdvMetroForm. Essa classe possui já a implementação do padrão minimalista. Além disso, adiciona novas propriedades para configurar sua aparência, a maioria delas está dentro da propriedade Appearence. Essas propriedades já são iniciadas com valores pré-definidos, por exemplo, o padrão de fonte é configurado para Segoe UI, que é o estilo utilizado no padrão Metro. Algumas dessas propriedades estão relacionadas a seguir.

- SystemIconColor: TColor : define a cor do estado normal dos botões de sistema localizados na barra de título;
- SystemIconColorHot: TColor : cor do estado mouse sobre para os botões de sistema localizados na barra de título;

• SystemIconColorDown: TColor : cor do estado pressionado para os botões de sistema localizados na barra de título;
• SystemIconColorDisabled: TColor : cor do estado desabilitado para os botões de sistema localizados na barra de título;
• Color: TColor : cor de fundo do formulário;
• SizeGripColor: TColor : cor do elemento de redimensionamento do formulário, localizado no canto inferior direito;
• CaptionFont: TFont: tipo de fonte utilizada no caption do formulário;
• CaptionColor: TColor : cor de fundo para o caption;
• CaptionStyle: TCaptionStyle : quando configurada para csMetro, o caption é dividido em uma parte localizada à esquerda, em uma cor diferente do restante do caption. Quando é configurada para csPlain, o caption é desenhado de forma se estender pela largura do formulário, desenhando uma barra de título completa;
• CaptionActiveColor: TColor : cor do caption quando o formuário está ativo;
• TextAlign: TAlignment : alinhamento do texto do caption;
• TextColor: TColor : cor empregada pela propriedade Text;
• Text: string: texto que é exibido no canto superior direito do caption do formulário.

Figura 13. Formulário Metro

O conceito Metro introduz um novo elemento visual chamado Tile. Eles representam aplicativos, páginas web, pastas no disco e outros itens. Em sua forma padrão são simples retângulos opacos, ou quadrados, que são organizados automaticamente em uma tela, formando algo como um grid. Em sua concepção original ao selecionar um Tile (seja por toque, mouse ou outro meio) o aplicativo ou recurso representado por ele é aberto. Usuários podem configurar esses Tiles de várias maneiras. É possível remover ou adicionar, movê-los para novas posições, criar ou modificar grupos de Tiles e muito mais. O mais interessante sobre os Tiles, é que foram projetados especialmente para exibir informação em tempo real de forma dinâmica, mesmo quando o aplicativo que ele representa não está em execução.

Com essa ideia em mente, as antigas opções do menu são bons candidatos a se tornarem Tiles. A TMS desenvolveu um componente Tile simples, é o TAdvMetroTile localizado na guia TMS Metro. Foram adicionados dois. Ambos tiveram os valores da sua propriedade Appearance modificadas. O valor para BorderColor e Color foi ajustado para $00F2BC00, que é um tom claro de azul. TextColor foi alterada para clWhite. Esses ajustes configuram a cor padrão quando não estão selecionados. Outra propriedade é a Picture, nela escolhemos uma imagem que será utilizada. Além disso também é possível definir um texto na propriedade Caption. Esse texto pode ser formatado utilizando tags HTML. Para a primeira Tile o Caption foi ajustado para "novo contato", enquanto que a segunda recebeu o texto "aniversariantes 3". A ideia dessa segunda Tile é que ao iniciar o sistema fosse realizada uma busca, uma contagem dos aniversariantes para o mês corrente. Isso agrega conteúdo. É uma pequena mudança que entra em conformidade com o Metro, focar no conteúdo. Assim, a **Figura 14** mostra o resultado obtido até o momento ao executar a aplicação.

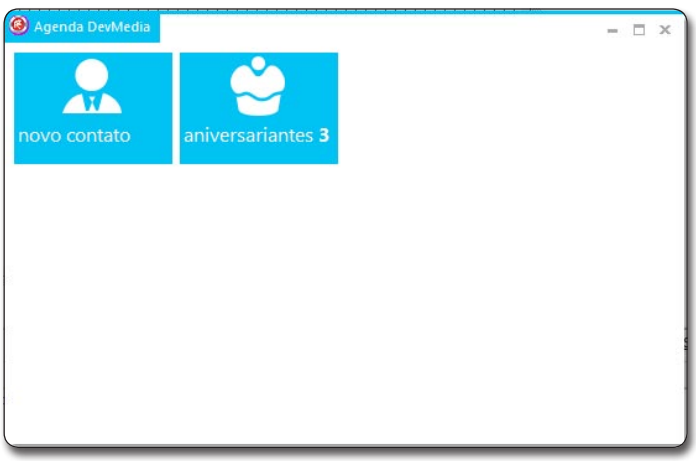


Figura 14. Tiles

Podemos melhorar ainda mais a usabilidade. Vamos permitir a pesquisa de contatos a partir do formulário principal. Então o usuário poderia excluir ou editar um contato selecionado.

Ao lado da Tile aniversariantes podemos acrescentar um Label, com fonte Segoe UI tamanho 12, com a mesma cor da Tile, contento o texto "Localize um contato". Abaixo dele adicionamos um TAdvEdit, da guia TMS Edits. Esse controle permite uma customização especial de sua aparência. Por exemplo, podemos setar como true sua propriedade FocusBorder para que sempre que o campo receber o foco, um destaque visual seja dado a ele. É possível configurar também a cor dessa borda. Isso é feito na propriedade FocusBorderColor, que foi configurada para a mesma cor do Label. Ao lado do campo foi acrescentando um TAdvMetroButton com sua propriedade Caption contendo "LOCALIZAR". Esse controle também é especializado no Metro, onde botões são representados por palavras completas. Abaixo dele foi posicionado um GroupBox que teve sua propriedade Visible configurada como false e dentro desse um DBGrid e mais dois botões TAdvMetroButton, agora com o caption "EXCLUIR"

e "EDITAR". Como estamos apenas simulando uma interface, o botão localizar não efetua busca alguma, ele irá verificar se existe um conteúdo digitando no campo de busca, se houver o GroupBox é exibido, se não, é escondido.

Além de controles visuais, o pacote TMS Metro também inclui funções de exibição de mensagens com uma aparência diferenciada. Essas funções estão localizadas nas units AdvMetroDlgs.pas e AdvMetroTaskDialog.pas, foram implementadas de tal forma que podem substituir funções originais da VCL e até mesmo do Windows.

Vamos adicionar essas duas units ao formulário principal e no clique do botão EXCLUIR vamos disparar uma confirmação utilizando o código a seguir:

```
if (MetroMessageBox(Handle,'Confirma a exclusão do registro?','Atenção', MB_YESNO + MB_ICONQUESTION)= ID_YES) then
   MetroShowMessage('Você escolheu apagar o registro');
```

Como pode ser visto, a utilização é a mesma das APIs existentes no Windows, a diferença fica por conta do nome da função que possui o sufixo "Metro" associado. A seguir temos as funções originais e como elas foram implementadas para garantir compatibilidade.

Procedure ShowMessage pode ser trocada por:

```
procedure MetroShowMessage(Text: string); overload;
procedure MetroShowMessage(Text, Caption: string); overload;
procedure MetroShowMessage(Text, Caption, CaptionText: string);
overload;
```

ShowMessageFmt pode ser substituída por:

```
procedure MetroShowMessageFmt(const Instruction: string; Parameters:
array of const);
```

MessageDlg pode ser substituída por:

```
function MetroMessageDlg(const Instruction: string; DlgType: TMsgDlgType;  Buttons:
TMsgDlgButtons; HelpCtx: Longint): Integer; overload;
function MetroMessageDlg(const Instruction: string; DlgType: TMsgDlgType;  Buttons:
TMsgDlgButtons; HelpCtx: Longint; DefaultButton: TMsgDlgBtn): Integer; overload;
```

A API MessageBox do Windows pode ser substituída por:

```
function MetroMessageBox(hWnd: HWND; lpInstruction, lpTitle: PChar; flags: UINT): Integer;
```

InputQueryDlg pode ser trocada por:

```
function MetroInputQueryDlg(ACaption, APrompt: string; var Value: string): boolean;
```

Para recriarmos a tela de inclusão de contatos temos que retirar agora algumas funcionalidades, visto que a consulta e exclusão estão centradas no formulário principal. Para adicionar um novo formulário TMS Metro acessamos o menu File>New>Other>TMS Forms >TMS Metro Form. Ele foi configurado como mostra a **Figura 15**.

Foram adicionados controles já utilizados até agora, como TAdvEdit e TAdvMetroButton que teve duas propriedades alteradas. A primeira delas é a propriedade Picture, que permitiu escolher uma imagem para o botão, a segunda é a ShowCircle, que quando habilitada desenha automaticamente um círculo em torno da imagem escolhida.

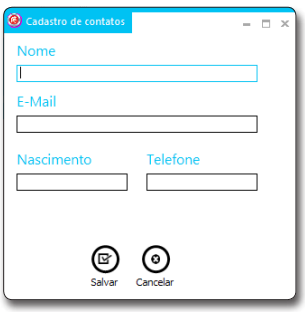


Figura 15. Cadastro de contatos

Conclusão

Talvez, muitos não tenham gostado do visual apresentado. Mas esse é um caminho sem volta nas aplicações Windows. Basta realizar uma consulta sobre o Windows 8 e vários exemplos aparecem. Mas acredito que a maior dificuldade não é mudar um visual, e sim, mudar o pensamento para criarmos interfaces que explorem mais conteúdo.

Com algumas mudanças simples de layout melhoramos a usabilidade, tornando mais claro o funcionamento do sistema para alguém que nunca o tenha utilizado. Como vocês perceberam, eu não implementei as mudanças para a tela de consulta de aniversariantes, deixo esse desafio para vocês, de pensar em algo inovador. Para ajudar no entendimento, a Microsoft disponibiliza alguns guias, estes vão mais a fundo abordando até mesmo a posição correta de labels e campos, o espaçamento recomendado, o tamanho recomendado.

Paulo Quicoli
pauloquicoli@gmail.com
Editor da revista ClubeDelphi e editor técnico .Net Magazine. Formado em processamento de dados pela FATEC-TQ. Atua como analista programador de sistemas e coordenador técnico da equipe de desenvolvimento .NET da Siplan Control-M (www.siplancontrolm.com.br) unidade Jaboticabal, e professor da FATEC, campus Taquaritinga. Blog: http://quicoli.worpress.com

TMS Componente Pack
http://www.tmssoftware.com/site/tmspack.asp

Microsoft guideline for Metro Style
http://msdn.microsoft.com/en-us/library/windows/apps/hh465424

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

ER/Studio Software Architect

Um overview sobre a ferramenta de modelagem de aplicações da Embarcadero

Resumo DevMan

De que se trata o artigo:

O artigo irá abordar o produto da Embarcadero voltado à modelagem de aplicações denominado ER/Studio Software Architect. Assim como os já apresentados ER/Studio Data Architect e ER/Studio Business Architect, ele é parte integrante do pacote ER/Studio XE2. Mantendo a intenção dos artigos anteriores, será apresentado um apanhado geral da ferramenta, desde seus conceitos básicos, até sua amostragem prática. O público alvo a ser atingido é a própria comunidade Delphi, fazendo com que seus profissionais tomem conhecimento de mais esta opção que pode contribuir e agregar valor à sua rotina de trabalho.

Em que situação o tema é útil:

Modelar um Software é uma atividade que envolve a elaboração de modelos que fundamentem suas características e comportamentos. Comumente, este tipo de atividade é tratada como Modelagem de Software, sendo utilizada principalmente com o intuito de se identificar os recursos e funcionalidades que o Software que está sendo construído deverá prover. Com o intuito de facilitar o processo de modelagem, frequentemente é utilizado algum tipo de notação gráfica, tal como a proporcionada pela linguagem gráfica UML (Unified Modeling Language). Como será visto, o ER/Studio Software Architect prevê o suporte à UML 2.0.

ER/Studio Software Architect:

Num primeiro momento, tem-se a falsa impressão de que a família XE2, apresentada no ano passado ao grande público pela Embarcadero, trouxe somente o Delphi e suas vertentes (RadPHP e Prism) como novidades. No entanto, não é necessário olhar muito a fundo para constatar a presença de outros produtos. Um deles é o ER/Studio XE2, que se apresenta como um pacote contendo três outros produtos, todos voltados à modelagem das mais variadas áreas, tais como dados, aplicações e negócios. Dentre estes, apresenta-se então o ER/Studio Software Architect, que é uma ferramenta de modelagem de Softwares orientados a objetos. Sua utilidade se estabelece em analisar e projetar, de maneira totalmente visual, softwares de toda a natureza, do mais trivial ao complexo. Um dos seus principais atrativos é o apoio à especificação UML 2.0, o que simplifica a fase conceptiva do projeto. O artigo tem então como finalidade apresentar um overview desta e das demais características da ferramenta, se apoiando sempre que possível em situações e esboços práticos.

Este artigo faz parte de um curso

O ER/Studio XE2 é a solução final da Embarcadero para a modelagem de aplicações, dados (data) e negócios (business). Sendo assim, ele contempla nada menos que três produtos distintos: ER/Studio Software Architect, ER/Studio Data Architect e ER/Studio Business Architect. Na prática, quando se realiza o download do ER/Studio XE2 diretamente do site da fabricante, você estará baixando um único pacote contendo três instaladores, cada qual condizente aos produtos mencionados. Pela simples razão destes tipos de ferramentas poderem agregar valor ao trabalho de um desenvolvedor Delphi, e com o acréscimo de todas serem da mesma empresa fabricante, foi proposta desde edições anteriores a abordagem de cada um destes produtos que formam o ER/Studio XE2. O leitor que acompanha mais de perto a Clube Delphi já pôde conferir de perto algumas das nuances do ER/Studio Data Architect e do ER/Studio Business Architect, além de pontuar a utilidade de cada uma em seu dia-a-dia de trabalho. Para finalizar o assunto, o presente artigo apresenta um overview sobre o último produto da tríade, o ER/Studio Software Architect. Até mesmo para simplificar o entendimento, daqui em diante ele será tratado por muitas vezes apenas como ER/Studio SA.

Dentre todos os três produtos, o ER/Studio SA pode ser considerado o caçula, estando ainda em sua versão de número 1 (mais

especificamente versão 1.1.0, até o momento deste artigo), o que não afeta em nada sua qualidade e eficiência. A fim de conceitos, podemos defini-lo como sendo um ambiente orientado a projeto (Design-Driven), voltado exclusivamente à modelagem de aplicações. Em vista disso, dentre seus principais recursos estão o suporte a padrões (Patterns), UML (Unified Modeling Language), OCL (Object Constraint Language), métricas e auditorias de Quality Assurance e geração automática de documentação. Alguns destes aspectos serão abordados em mais detalhes no decorrer do artigo.

A UML

Podemos definir a UML como uma linguagem de modelagem, provida com a intenção de padronizar as formas de modelagem de um software. Dentre seus principais objetivos estão a especificação e documentação de toda a logística envolvida no desenvolvimento de determinado sistema de informação. Ainda sobre conceitos, é importante dizer que a UML não irá traçar os passos sequenciais para se projetar um sistema, uma vez que ela não é uma metodologia de desenvolvimento. Ao invés disso, ela irá prestar todo o auxílio para melhor visualização do desenho estrutural e lógico dos objetos do sistema. Olhando pelo lado do desenvolvedor em si, um dos grandes atrativos da UML é a visualização de seus trabalhos através de diagramas, relacionados a cada bloco de construção provido por ela. Em vista do grande número de diagramas disponíveis, eles são categorizados de acordo com seu contexto, tais como de estrutura (estruturais), de comportamento (comportamentais) e interação. Diagrama de Classe, Diagrama de Componentes, Diagrama de Caso de Uso e Diagrama de Sequência são alguns exemplos tradicionais de diagramas e todos eles estão disponíveis no cenário do ER/Studio SA, como será visto no decorrer do artigo.

Nota do DevMan

OCL é a sigla utilizada para o termo Object Constraint Language que, traduzido livremente para o português, seria algo como Linguagem para Especificação de Restrições em Objetos. Seu conceito está intimamente ligado à UML, por se tratar de uma linguagem utilizada para se descrever regras a serem aplicadas a seus modelos. Na prática, através de uma linguagem totalmente textual, permite a expressão de restrições não especificadas pelos diagramas, a um modelo orientado a objeto.

Usabilidade

Apesar de não ser regra, na grande maioria das vezes uma nova ferramenta quando apresentada, causa certa desconfiança a seus futuros candidatos a utilizadores, até que seu uso seja justificado e devidamente compreendido. Conforme dito, a premissa do ER/Studio SA é prover recursos para a modelagem de software. Com isso em mente e com base no que será apresentado, podemos rapidamente estipular que uma empresa ou corporação poderá melhorar seus processos de definição de requisitos, utilizando a notação de modelagem UML. Isso contribui diretamente para que as aplicações sejam então projetadas e desenvolvidas, primando por itens como reusabilidade, unicidade e coesão. O software torna-se então adaptável a várias situações, podendo ser bem compreendido em qualquer circunstância e por diferentes envolvidos.

Ao citar o tema Usabilidade, alguns outros tipos de profissionais, além dos desenvolvedores, também poderão tirar proveito do ER/Studio Software Architect. Um exemplo clássico são os analistas, que poderão basear suas análises de requisitos nos diagramas de caso de uso UML produzidos. Na mesma medida, arquitetos poderão introduzir melhorias nas definições da modelagem do software propriamente dita, através da introdução de métricas e auditorias que irão monitorar o nível de qualidade do que está sendo produzido.

Instalação

O processo de instalação do ER/Studio SA segue o mesmo molde adotado pela maioria das ferramentas disponibilizadas pela Embarcadero, no mais tradicional Wizard (Next-Next-Finish) de aplicações Windows. Sendo assim, não cabe dispender algum tempo para explicar algo tão trivial. Uma vez instalado, o IDE da ferramenta torna-se disponível para uso.

IDE

O termo IDE é o acrônimo para Integrated Development Enviroment, que numa tradução simples para o português seria algo como Ambiente Integrado de Desenvolvimento. Ele é utilizado para designar uma ferramenta que reúna recursos e características que apoiem e, consequentemente, agilizem o desenvolvimento de software. Sendo assim, o IDE do ER/Studio Software Architect é formado por componentes visuais, tais como modelos, editores, diagramas, paletas e visualizadores de propriedades, que proporcionam toda sua produtividade na melhor forma RAD, assim como acontece no Delphi. Para fins de conhecimento, a **Figura 1** ilustra uma visão inicial do ambiente. Como pode ser observado, assim como acontece em alguns dos produtos da Embarcadero, o ER/Studio SA é todo baseado em outro ambiente de desenvolvimento, o Eclipse, muito conhecido e utilizado no nicho Java.

Ainda sobre seu IDE, o ER/Studio SA faz uso de duas estruturas básicas no que diz respeito à organização e exibição das informações envolvidas: Workspace e Project. Um Workspace, ou espaço de trabalho, é um contêiner para projetos. Sendo assim, numa hipotética estrutura Treeview ele seria o elemento de nível superior (Top level), podendo conter um ou N projetos filhos. Isso significa dizer também que apenas um Workspace poderá estar aberto por vez no IDE. Em termos visuais, considere um Workspace o próprio ambiente de trabalho do IDE como um todo. Já um Project, ou projeto, é uma estrutura lógica que irá conter todos os recursos necessários a determinado trabalho. Isto porque, todo e qualquer trabalho realizado no ER/Studio SA é realizado e efetivado dentro do contexto de um projeto. Toda a estrutura formada por um ou N Projects é disposta no IDE numa guia denominada Model Navigator View, conforme mostra a **Figura 2**.

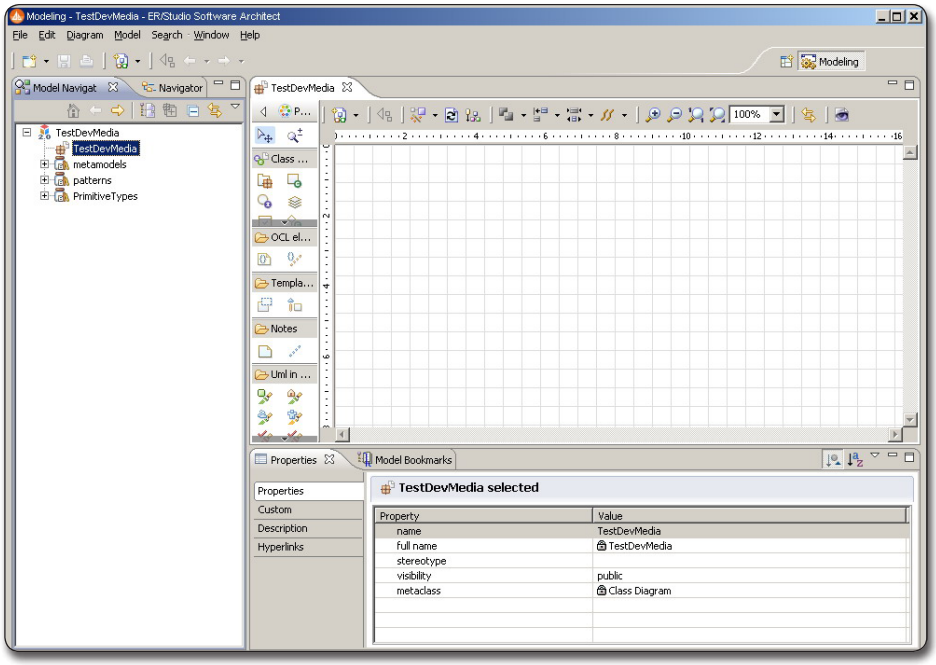


Figura 1. IDE do ER/Studio Software Architect

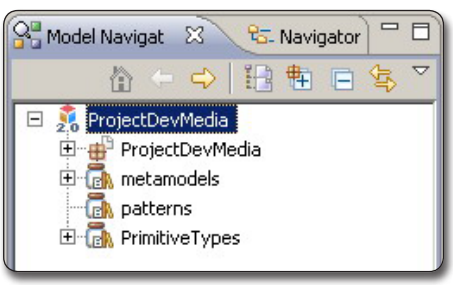


Figura 2. Model Navigator View

No ER/Studio SA um projeto pode ser de vários tipos, mais precisamente seis, todos relacionados direta ou indiretamente à modelagem:

- UML 2.0: projeto com suporte exclusivo à UML 2.0 e toda sua parte visual, sem apoio a código-fonte;
- Pattern Definition: projeto que permite a criação de novos padrões (Patterns) de modelagem baseados na UML 2.0;
- Profile Definition: projeto que permite a criação de novos perfis de modelagem;
- Packages: determina a criação de pacotes, utilizados principalmente por fins de organização dos modelos de um projeto;
- Diagram: determina a criação de um diagrama, que são estruturas que irão conter os elementos gráficos utilizados para representar cada parte do modelo;
- Model Elements: determina a criação de elementos do modelo;

Ainda falando sobre Packages, ou pacotes, eles possuem duas representações para o contexto: lógico e físico. Olhando pelo lado lógico, cada modelo manipulado no ER/Studio SA é considerado um pacote e que pode ser constituído de um ou mais pacotes. Um pacote torna-se então um elemento do modelo utilizado para agrupar outros elementos, agindo com uma espécie de namespace. Fisicamente falando, um pacote representa em disco uma pasta contendo arquivos, os quais armazenam os eventuais diagramas e elementos de modelo de um projeto. Em vista da provável confusão inicial que possa se fazer sobre esta relação entre pacotes e modelos, é utilizada uma nomenclatura padrão para tratar certos elementos. Dessa forma, o modelo comumente é o pacote raiz de um projeto, e é tratado como Pacote Padrão (Default Package). O diagrama deste mesmo pacote é considerado o Diagrama Padrão (Default Diagram).

Na prática, projetos de modelagem serão manipulados no IDE. Cada projeto deste poderá conter um conjunto de diagramas, que nada mais são que as representações gráficas de cada parte do modelo. Um diagrama pode assumir vários tipos, o que acaba por definir diretamente o tipo de elemento que poderá ser utilizado em sua composição. Em vista disso, o ER/Studio Software Architect apresenta uma paleta específica para cada tipo de diagrama. Class Diagram (Diagrama de Classe), Activity Diagram (Diagrama de Atividade) e Sequence Diagram (Diagrama de Sequência) são alguns exemplos de tipos de diagramas (no caso, da UML 2.0), providos pela ferramenta.

Sem deixar de mencionar, Model Elements são os elementos do modelo com o qual você irá manipular. Para ser mais claro, cada modelo presente em um projeto de modelagem é composto por um agregado de entidades. Uma entidade é uma instância de uma metaclasse relacionada a um metamodelo do projeto. Por fim, cada instância desta é considerada um elemento do modelo, que possui propriedades e notações próprias, definidas para sua Metaclasse. Em razão da notação gráfica provida por esses elementos, eles podem ser inseridos em diagramas.

Nota do DevMan

No desenvolvimento de softwares, Metaclasse é um termo empregado e relacionado à orientação a objetos. Similarmente a uma classe tradicional, que define o comportamento de um objeto, uma Metaclasse define o comportamento de determinada classe e suas instâncias. Conceitualmente falando, uma Metaclasse pode ser rapidamente definida como uma classe, cujas instâncias também são classes e não objetos.

Tutorial

Iniciando a modelagem de softwares

O início da modelagem de aplicações com o ER/Studio Software Architect tem como ponto de partida a indicação de um Workspace, onde serão armazenados seus futuros projetos. Olhando pelo lado físico, um Workspace nada mais é que uma pasta localizada no disco nomeada por padrão como Workspace, que irá conter os arquivos relacionados aos projetos manipulados em seu espaço de trabalho. Por padrão, a própria ferramenta já oferece um Workspace localizado em C:\Documents and Settings\[Usuário]\ERStudioSoftwareArchitect\

workspace. A **Figura 3** exibe a caixa de diálogo Workspace Launcher provida pelo ER/Studio SA, para a seleção de um Workspace para trabalho.

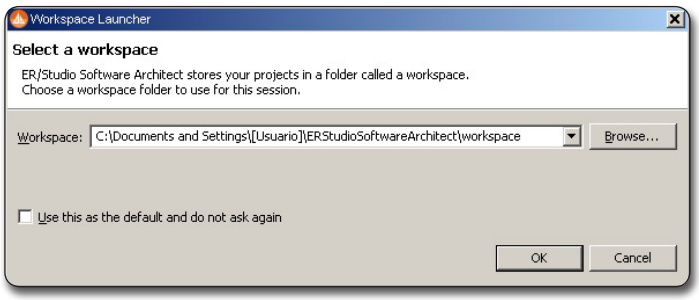


Figura 3. Selecionando um Workspace

A etapa seguinte é a criação do(s) projeto(s). Conforme já mencionado anteriormente, pelo fato da ferramenta ser focada na modelagem de aplicações orientada a objetos, o exemplo mais clássico e real de um projeto inicial é provavelmente a criação de um projeto UML 2.0. Intuitivamente a criação de um projeto no IDE segue os moldes do Delphi, estando esta opção disponível no menu File>New>UML 2.0 Project. Provando sua eficiência no que diz respeito aos seus recursos visuais, ao fazer isso, uma nova janela de diálogo é aberta, para apoiar o usuário na definição de algumas configurações. Entre estas está nomeação do novo projeto, bem como a indicação da localização de onde será salvo. Por padrão uma pasta relacionada é criada em disco, internamente à pasta do Workspace em que se está trabalhando. Em um projeto deste tipo, a integração de diagramas torna-se imprescindível, sendo assim, neste mesmo diálogo é possível indicar um diagrama inicial, tal como de Use Case, Sequence, entre outros. Ao final do processo, o novo projeto é disposto no Model Navigator View, dando-se início à modelagem propriamente dita.

Manipulando diagramas UML

Traçando um paralelo com a realidade, podemos estabelecer que a elaboração de um software orientado a objetos se relaciona diretamente com a modelagem UML e, consequentemente, o trabalho com seus diagramas. Dessa forma, no ER/Studio SA, uma vez criado um projeto deste tipo (UML 2.0 Project) o passo seguinte envolve a construção e manipulação destes diagramas. Elucidando um pouco este tipo de atividade no IDE, tomemos como base dois dos diagramas em projetos relacionados à UML, o de Classes e o de Caso de Uso.

Diagrama de Classes

Um Diagrama de Classes, ou Class Diagram como é tratado na ferramenta, é a representação gráfica de toda a estrutura de classes vinculadas ao contexto da aplicação que está sendo produzida. Vendo pelo lado do desenvolvimento propriamente dito, tais classes servirão de modelo para os futuros objetos utilizados no sistema. Sua importância ainda vai além, pelo fato do diagrama de classes ser a base para construção de outros diagramas. Entre seus conceitos relacionados, estão os elementos que irão compor efetivamente o diagrama. Conforme citado anteriormente, o IDE disponibiliza determinados e específicos elementos de acordo com o tipo de diagrama com que está se trabalhando. Assim sendo, para um Class Diagram os seguintes elementos estão disponíveis no menu de contexto, conforme mostra a **Figura 4**.

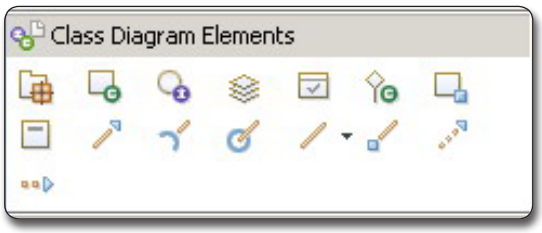


Figura 4. Elementos disponíveis a um Diagrama de Classes na ferramenta

Diante dos elementos mostrados, vale tecer um comentário sobre alguns, em razão de sua maior usabilidade e até mesmo pelo grau de importância no contexto. Por exemplo, uma Class (ícone retangular com a letra C na parte inferior), é o elemento que irá esboçar uma classe no diagrama. Classe esta que nada mais é que um elemento abstrato representativo de um conjunto de objetos. É nela que estarão as especificações e características de cada objeto que, pela perspectiva do desenvolvimento, serão seus atributos e métodos. Outro conceito relevante neste cenário são os tipos de relacionamentos estabelecidos entre as classes. Em suma, os possíveis relacionamentos são expressos na ferramenta pelos elementos cujos ícones possuem o formato de aste ou flecha. Agregação, Composição e Especialização (ou Generalização) são alguns destes tipos de relacionamentos.

Começando pela mais trivial, a Generalização (Generalization no IDE) é utilizada para representar uma relação de herança, demonstrando as possíveis hierarquias existentes entre as classes. Já Agregação e Composição são consideradas tipos especiais de associações. Enquanto a primeira expressa uma relação de todo e parte, em que um todo é composto de N partes, a Composição estabelece uma relação de contenção. Isso significa dizer que um determinado objeto assume a posição de uma espécie de container, podendo assim conter outros objetos. Na ferramenta, uma associação é representada pelo elemento Association.

Diagrama de Caso de Uso

Um Diagrama de Caso de Uso é encontrado na ferramenta sob a nomeação Use Case Diagram. Conceitualmente falando, em suma, ele irá descrever uma determinada funcionalidade prevista para o sistema que está sendo projetado. Para isso é utilizado um alto nível de abstração, uma vez que o objetivo principal é apenas expressar de forma clara e concisa de que forma elementos externos irão interagir com a(s) funcionalidade(s) do sistema. Em termos de conceito, neste contexto dois itens assumem uma posição de destaque. Em vista da sua importância na construção dos diagramas, tais itens são na verdade elementos, denominados Ator (Actor) e Caso de Uso (Use Case).

Uma vez inserido em um Use Case Diagram, em termos visuais, um Ator assume a forma de um boneco. Ele irá exercer então o papel de uma entidade externa que irá interagir com o sistema, podendo ser um usuário ou mesmo outro sistema. Já um Caso de Uso, especifica uma determinada ação ou sequência lógica executada pelo sistema. No diagrama é expressa sob a forma de uma elipse. A **Figura 5** mostra os elementos disponíveis para Use Case Diagram no ER/Studio Software Architect.

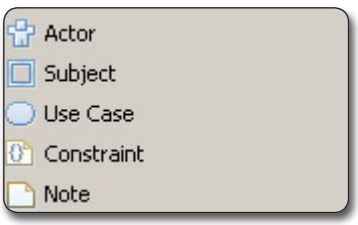


Figura 5. Elementos disponíveis a um Diagrama de Classes na ferramenta

Vendo pelo lado prático, o início da construção de um Diagrama de Caso de Uso no IDE, comumente se dá pela adição de um elemento Subject, que irá representar o sistema que está sendo tratado na ocasião. Posteriormente, entra em pauta a figura dos atores envolvidos na representação gráfica que está se montando, uma vez que se deve ter em mente o aspecto do sistema que se deseja demonstrar e, portanto, quem irá interagir com a mesmo. Como complemento, define-se então os Casos de Uso envolvidos e as possíveis associações existentes entre os elementos (atores e casos de uso). A **Figura 6** mostra um exemplo simples de um Diagrama de Caso de Uso construído na ferramenta, onde podem ser observados os elementos mencionados até aqui.

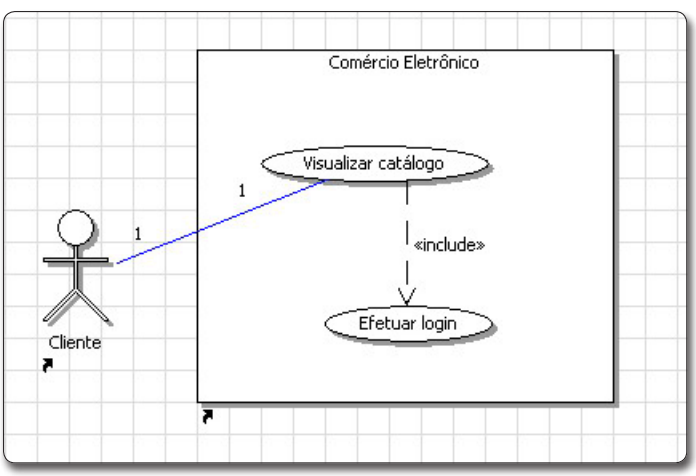


Figura 6. Exemplo de Use Case Diagram

Foi definida uma relação entre os casos de usos envolvidos. Isto dá margem para fazer menção a outro conceito importante no contexto dos Diagramas de Casos de Uso que são os possíveis tipos de relações existentes entre estes elementos. Relações estas que determinam a forma de interação entre os Casos de Uso. Há basicamente duas formas de relações diferentes. A primeira delas, denominada Include (e que foi a utilizada no exemplo), conforme sua nomenclatura supõe, estipula uma interação onde um Caso de Uso (Caso de Uso 1) pode incluir outro (Caso de Uso 2), de tal forma que se estabeleça uma relação de obrigatoriedade. Sendo assim, o comportamento do Caso de Uso 2 é incluído no Caso de Uso 1, ou seja, sempre que a ação de 1 ocorrer a de 2 também ocorrerá. Posto isso, poderíamos traduzir a **Figura 6** como "um cliente deverá efetuar o login ao sistema para que possa visualizar o catálogo". A segunda relação possível é denominada Extend, em que um Caso de Uso pode estender outro. Ao contrário do Include não se estabelece uma relação de obrigatoriedade, uma vez que a ação do Caso de Uso estendido não necessariamente será inserida no Caso de Uso extensor. No IDE, estas interações são explicitamente marcadas como <<include>> e <<exclude>> no diagrama, assim como acontece na maioria das ferramentas do gênero.

Ferramentas auxiliares

Assim como acontece nas demais ferramentas da família ER/Studio XE2 da Embarcadero, o ER/Studio Software Architect provê ao profissional uma série de ferramentas auxiliares que visam o aumento da produtividade de trabalho. Esse tipo de recurso vai se tornando útil na medida em que a utilização da ferramenta se torna uma constante no dia-a-dia de seu utilizador. Popularmente falando, podemos dizer que tais recursos auxiliares não são essenciais para a vida útil do produto, mas pelo fato de estarem disponíveis, faz com que se tenha um ganho de produção. Adicionalmente, algumas dificuldades ocasionais surgidas também são naturalmente sanadas. Elucidando alguns casos de exemplo, podemos ter situações em que determinado projeto venha a possuir vários modelos agregados, cada qual contendo vários objetos. Neste caso, certamente para o seu criador, localizar um modelo ou elemento específico é trabalho trivial, sem grandes complicações. Todavia para uma pessoa que observa o projeto pela primeira vez, essa simples tarefa pode acabar por se tornar algo dificultoso. Em vista disso, uma solução viável seria a presença de algum recurso capaz de prover uma busca por modelos, baseada em algum parâmetro indicativo. Outro caso ainda seria um modelo de grande porte, contendo inúmeros elementos. Em situações como esta, muitas das vezes a grande dificuldade encontrada diz respeito à visualização dos elementos e do todo, em virtude do grande número de itens espalhados pela tela. Uma solução plausível seria então a disponibilidade de um recurso adicional para auxiliar na navegação e na própria visualização destes elementos.

Search Tool

Search é o nome de uma das ferramentas disponibilizadas pelo ER/Studio SA. Como seu próprio nome sugere, trata-se de uma ferramenta de pesquisa que, na prática, realiza uma varredura pelo Workspace como um todo, a fim de encontrar o objeto que se

procura. Entenda por objeto qualquer item presente nos projetos de um Workspace, tais como modelos, diagramas e elementos. No IDE, este recurso fica localizado no menu homônimo Search (Search>Model). Uma vez acionado, é provida ao usuário uma janela de diálogo conforme mostrada na **Figura 7**.

Pela imagem, podemos ver o que a ferramenta nos dá de opções para a realização da pesquisa, como o parâmetro (Search String) a ser utilizado, se a busca será Case Sensitive ou não. Além disso, pode-se procurar por um determinado nome ou qualquer outra propriedade de um Modelo. Em termos de escopo, por padrão o próprio Workspace é o utilizado. No caso apresentado, foi utilizado como parâmetro de pesquisa a string UseCase, determinada ao escopo do Workspace em questão. Tão logo, o resultado do processo é mostrado numa guia específica, Search, no rodapé da IDE (**Figura 8**).

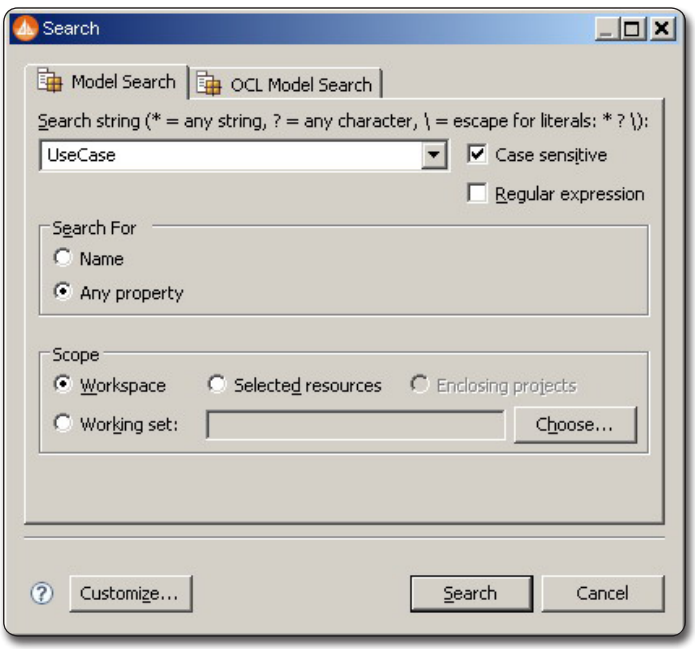


Figura 7. Search Tool

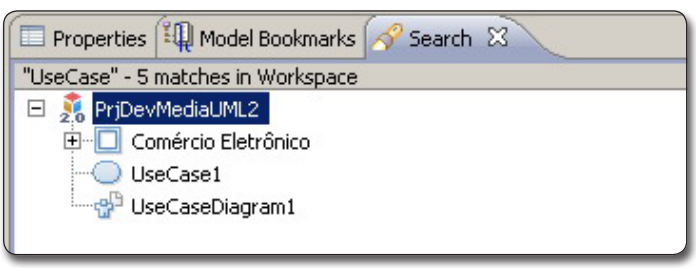


Figura 8. Search – Guia de resultados

Zoom Tool

Talvez a mais trivial de todas as ferramentas disponíveis na IDE, o Zoom é um recurso presente na maioria dos ambientes que tratam de formas gráficas, possibilitando o usuário aumentar ou diminuir a perspectiva de visualização. Dito isto, no ER/Studio Software Architect a razão principal pela presença de um recurso como este é pelo seu trabalho com Diagramas. Sendo assim, estruturas com diversos elementos podem ter um Zoom diminuído para facilitar a visualização do cenário como um todo, ao mesmo tempo em que de maneira oposta, um Zoom aumentado facilita a visualização de blocos e regiões específicas.

No IDE, o recurso de Zoom é provido fisicamente em dois lugares distintos, mas ambos direcionados para o mesmo contexto, que é o Diagrama em si. O primeiro local é refletido no menu Diagrams, enquanto que o segundo é na própria barra de ferramentas (Toolbar) de uma diagrama. As **Figuras 9** e **10** ilustram estes dois locais.

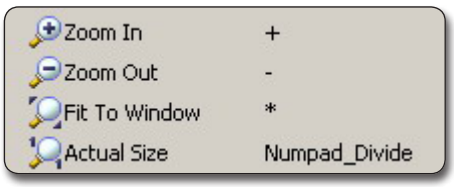


Figura 9. Zoom localizado no menu Diagram

Figura 10. Zoom localizado na toolbar de um diagrama

Conforme pôde ser visto, o recurso de Zoom apresenta alguns valores pré-determinados, que visam facilitar sua utilização:

- Zoom In: como seu próprio nome sugere, a cada clique, aumenta-se o percentual de visão;
- Zoom Out: como seu próprio nome sugere, a cada clique, diminui-se o percentual de visão;
- Fit To Window: o nível de Zoom é reduzido até que todo o diagrama se encaixe devidamente na área visual;
- Actual Size: funciona como uma espécie de Reset, retornando o diagrama ao seu tamanho original do momento de sua criação.
- Combo de porcentagem: disponível na toolbar do diagrama, permite ampliá-lo ou reduzi-lo com base nos percentuais pré-definidos.

Overview Window

Em termos funcionais, o recurso de Overview pode ser utilizado em diagramas de grande proporção, que invariavelmente acabam por conter uma grande quantidade de elementos. Nestes casos, a navegação acaba sendo automaticamente prejudicada, pelo fato do espaço físico do próprio diagrama extrapolar a área visual da tela. Para contornar esta dificuldade, o Overview aciona uma espécie de miniatura da visão do modelo, contendo uma área sombreada que provê a você um rápido deslocamento por toda a área do modelo.

No IDE, o recurso de Overview pode ser acionado tanto pela toolbar, quanto pelo menu Diagram, através de um ícone característico (**Figura 11**), nomeado como Show Overview. A **Figura 12** mostra então um Diagrama Caso de Uso com o recurso de Overview acionado.

Figura 11. Overview

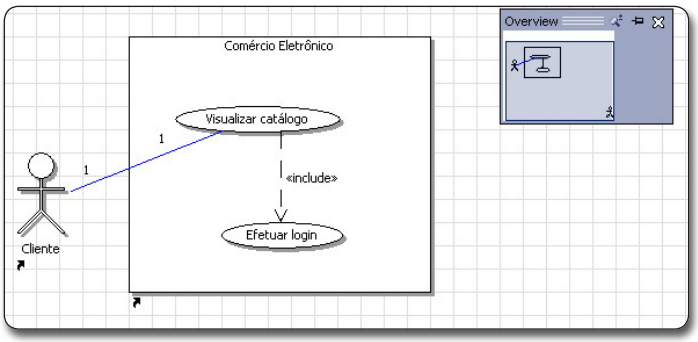


Figura 12. Overview em ação

Commands

Ainda relacionado a diagramas, há quatro recursos especiais disponibilizados voltados à sua personalização. Pelo lado conceitual, eles são tratados como comandos (Commands). O primeiro deles é então nomeado como Grid (menu Diagram>Grid) e, basicamente, exibe linhas no formato de grade na visão do diagrama. Este comando vem acionado por padrão, conforme pode ser confirmado na imagem do Diagrama de Caso de Uso mostrado anteriormente. Sua funcionalidade principal é contribuir na orientação visual dos elementos que compõem o modelo.

Outro comando disponível é denominado Align (menu Diagram>Align). Como facilmente pudemos supor, ele está diretamente relacionado ao alinhamento dos elementos do diagrama, trabalhando com opções pré-definidas pelo IDE. Como exemplo, temos Align Top, Align Left e Align Right, que alinham os elementos ao topo, à esquerda, à direita, respectivamente. Ainda nesta mesma linha direcionada ao layout do modelo, outra ferramenta disponível apresenta-se sob o intuitivo nome de Layout (Diagram>Layout). Em suma, este recurso manipula a disposição dos elementos presentes em um diagrama. Para este processo, ele faz uso de opções pré-determinadas, tais como Layout All for Printing, Layout All e Layout Selected Elements. Tomando como base a primeira opção, Layout All for Printing, na prática ela então irá definir uma disposição para todos os elementos envolvidos usando um algoritmo específico destinado à impressão. De forma semelhante, Layout All irá dispor os elementos de acordo com o algoritmo de layout definido. Nativamente, esta definição do algoritmo a ser utilizado é feita através do menu Preferences (Windows>Preferences) da ferramenta, conforme mostra a **Figura 13**.

Por fim, o último Command disponível é o denominado Match (Diagram>Match), que serve para combinar a altura ou largura dos elementos do diagrama. Tome como exemplo os casos onde você necessite, por questões de estética, estabelecer uma altura ou largura padrão a um grupo de elementos selecionados. Sendo assim, ao acionar o comando Match, você poderá igualar seus valores para Width (largura) e Height (altura). Até mesmo por esta razão, o recurso trabalha com duas opções fixas: Match Width e Match Height.

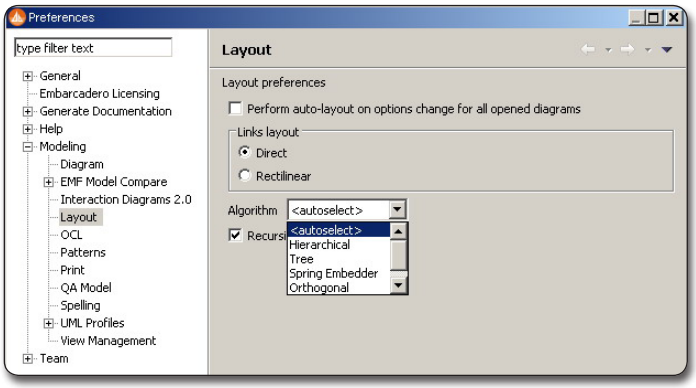


Figura 13. Layout Algorithm

Conclusão

Olhando pelo lado do desenvolvedor, o presente artigo teve como principal objetivo expor mais uma das ferramentas da Embarcadero complementares ao Delphi. Dessa forma, fechamos uma série de artigos que apresentou os três produtos da solução ER/Studio XE2: ER/Studio Data Architect, ER/Studio Business Architect e ER/Studio Software Architect.

Fazendo um balanço de tudo que foi apresentado, podemos perceber que estamos bem servidos no que diz respeito a ferramentas auxiliares ao desenvolvimento. Prova disso é que esta tríade de produtos provê soluções para a modelagem de dados, modelagem de negócios e modelagem de Software.

Fabrício Hissao Kawata
fabricio.kawata@bol.com.br
Formado em Processamento de Dados pela FATEC-TQ. Atua como Analista Programador Delphi há 7 anos. Atualmente escreve artigos sobre Delphi para revistas especializadas.
Blog: prismsilver.blogspot.com

Embarcadero ER/Studio XE2 – Página de download
https://downloads.embarcadero.com/free/er-studio

Embarcadero ER/Studio Software Architect – Página oficial do produto
http://embarcadero.com/products/er-studio-software-architect

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Desenvolvendo para Android - Parte 3

RadPHP e Delphi XE 2

Este artigo faz parte de um curso

Resumo DevMan

De que se trata o artigo:

Esta terceira parte do minicurso tem o objetivo de iniciar o desenvolvimento da aplicação Cliente que vai consumir o servidor de aplicação DataSnap apresentado na edição anterior, através de um Software para o sistema operacional móvel Android que também pode ser utilizado para o iOS, desenvolvido com a ferramenta RadPHP XE 2 em conjunto com a PhoneGap, uma API livre que permite o emprego das tecnologias presentes na Web (WebStandards) para o desenvolvimento de aplicações para diversas plataformas.

Em que situação o tema é útil:

Desenvolver um software de forma ágil e simplificada para os sistemas Android e/ou iOS utilizando diversas como HTML, CSS e JavaScript, assim como o Framework jQuery e a API PhoneGap.

Desenvolvendo para Android:

O mercado móvel se mostra cada vez mais atrativo. São softwares, jogos, Internet, players de músicas, vídeos, diversos recursos que cabem "na palma da mão" e fazem com que os smartphones se tornem minicomputadores de bolso. Evidentemente, com o grande crescimento deste mercado, o desenvolvimento de aplicações para esta plataforma se torna um ponto importante e fundamental para uma organização que deseja inovar e oferecer mais recursos aos seus clientes. Dentro deste contexto, o Delphi em conjunto com a ferramenta RadPHP, permite a concepção de softwares que trabalham de forma integrada e aumentam assim as possibilidades da empresa. Abordamos neste artigo as principais tecnologias da Web empregadas no desenvolvimento de páginas (HTML e JavaScript) para a elaboração da aplicação Cliente Android com o auxílio da API PhoneGap, que por sua vez, permite o desenvolvimento de aplicações que podem ser exportadas para diversas plataformas.

Uma boa alternativa para responder ao crescimento da mobilidade é incorporar nas aplicações comerciais módulos que podem ser utilizados nos dispositivos móveis, aumentando assim a acessibilidade e praticidade dos mesmos, além evidentemente de ser um importante diferencial competitivo. Após o desenvolvimento do servidor de aplicação DataSnap, através do Delphi XE 2, que oferece a base para a realização de uma venda através de um dispositivo móvel, assim como a aplicação WebApp teste para consumir e testar a conectividade, o próximo passo é iniciar a elaboração da aplicação Cliente, que deve se executada diretamente no dispositivo móvel baseado no S.O. Android. Neste momento, o RadPHP XE 2 se torna um grande aliado para o desenvolvimento de softwares para outras plataformas pois além de utilizar a linguagem de programação PHP como base, oferece suporte a outras tecnologias como o HTML 5, biblioteca JavaScript jQuery e à API PhoneGap, além evidentemente dos recursos em tempo de Design. Diante deste fator, para utilizar todos os recursos que o RadPHP oferece, é necessária um certo conhecimento nestas linguagens e ferramentas.

A linguagem HTML

O HTML (HyperText Markup Language) é uma linguagem de marcação padrão base utilizada para a elaboração de páginas da Internet que nasceu essencialmente de dois padrões, o HyTime (Hypermedia/Time-based Document Structuring Language) e o SGML (Standard Generalized Markup Language). O padrão HyTime é utilizado para representar estruturalmente uma informação ou hipermídia com base em tempo, oferecendo os recursos necessários para a construção padronizada de sistemas baseados em hipertextos. Já o padrão SGML é muito conveniente para a transformação de documentos em hiper-objetos e para descrição de informações, apesar de não ter sido elaborado para hipertexto. Neste contexto também está presente o padrão DTD (Document Type Definition), que é utilizado para a definição das regras de formatação que devem ser empregadas em determinada classe de documentos. A linguagem HTML evolui constantemente e atualmente está na versão 5, sendo uma linguagem interpretada pelo navegador e que pode ser escrita através de qualquer editor de textos. A sua base são as tags (etiquetas), que por sua vez são utilizadas para descrever

todo o conteúdo de uma página e são definidas com a utilização dos caracteres "<" e ">". A maior parte das tags deve possuir outra tag correspondente para o fechamento da mesma, marcando assim onde é o início e o fim de determinado elemento, bem como tudo que está relacionado a ele. Também é importante salientar que as tags não são case-sensitive, portanto, podem ser escritas com letras maiúsculas ou minúsculas. Como o RadPHP XE 2 também se baseia nesta linguagem, é muito importante conhecê-la para manipulá-la e usufruir de todos seus recursos.

Uma página HTML tem como base, a anatomia apresentada a seguir:

```
<DOCTYPE>
<html>
 <head>
  { Preâmbulo - Informações do cabeçalho }
 </head>
 <body>
  { Conteúdo da página }
 </body>
</html>
```

Como é possível observar, as tags <html> e </html> são as delimitadoras para a descrição da página. No preâmbulo (<head>), são definidos valores como o título da página, as meta-informações, os estilos, os scripts e etc. O corpo de uma página, delimitado pelos marcadores <body> e </body>, deve conter os elementos HTML e todas as informações que serão apresentadas para o usuário, em outras palavras, o conteúdo visível, tais como textos, imagens, formulários, animações, vídeos, sons, entre outros. A **Tabela 1** apresenta as principais tags para elementos HTML que também serão utilizados neste exemplo, além das já mencionadas anteriormente.

Outro ponto extremamente importante na linguagem HTML é a definição dos atributos para os elementos. Dentro deste contexto, um atributo que merece destaque é o "id", uma vez que o mesmo possibilita uma melhor integração entre os elementos com as outras tecnologias envolvidas, como é o caso do JavaScript. O RadPHP insere automaticamente este atributo com base no valor presente na propriedade "name" de um objeto. A **Tabela 2** apresenta alguns importantes atributos que podem ser definidos em grande parte dos elementos.

Nota do DevMan

Meta-informações são dados adicionais que são inseridos em páginas HTML cujo objetivo principal é descrever várias informações sobre o documento, tais como palavras-chave lidas principalmente pelos mecanismos de busca, sendo uma informação extremamente importante em um site presente na Web. Uma meta-informação pode ser definida com o uso da tag de marcação <meta> seguida por um atributo e seu conteúdo. Uma meta tag também pode ser aplicada para redirecionar a exibição de uma página, atualizá-la automaticamente após algum tempo, dentre outros recursos. Ao desenvolver uma aplicação baseada em páginas HTML, como é o caso deste projeto, o emprego deste recurso não é uma obrigatoriedade.

Nota do DevMan

O CSS (Cascading Style Sheets – folhas de estilos em cascata) é uma importante tecnologia que foi introduzida na linguagem HTML a partir da versão 3.2, que permite a formatação uniforme de todas as páginas de um site, possibilitando a personalização dos elementos, das classes, dentre outros recursos. Uma vez que estas informações se encontram centralizadas em um documento CSS, a manutenção e alteração da formatação do conteúdo que faz referência à mesma torna-se muito simples e rápida.

JavaScript e o Framework jQuery

A linguagem de script Client-Side JavaScript foi concebida por Brendan Eich da Netscape, sendo baseada no padrão ECMAScript, e se tornou uma das principais e mais populares tecnologias empregadas no desenvolvimento de WebSites, uma vez que é suportada por uma diversidade de navegadores. Foi desenvolvida

Tag ou elemento	Descrição
<title>Título</title>	Definição de um título para a página.
<script>...</script>	Utilizada para inserir um script no documento HTML.
<style>...</style>	Utilizada para a definição de classes e estilos com a utilização de CSS para os elementos do HTML.
...	Permite definir uma divisão ou seção em um documento HTML, sendo um dos elementos mais importantes na elaboração do conteúdo.
<table>...</table>	Definição de uma tabela.
<tr>...</tr>	Definição de uma linha de células para uma tabela.
<td>...</td>	Definição de uma célula para uma tabela.
<a>...	Ligações âncora, permitindo a definição de um hiperlink e, consequentemente, a troca entre páginas.

Tabela 1. Principais tags da linguagem HTML

Atributo	Descrição
id	Utilizado para a identificação de determinado elemento.
name	Permite a definição de um nome para o elemento.
class	Permite a definição de uma classe para o elemento em questão.
style	Permite a atribuição de um ou mais estilos de forma direta no elemento em questão.

Tabela 2. Atributos importantes para os elementos da linguagem HTML

com o objetivo principal de permitir uma maior interatividade em páginas HTML, oferecendo várias funcionalidades como eventos, animações, validações, manipulação de conteúdo, comunicação com servidores externos, obtenção de informações do navegador cliente e etc. Como seu próprio nome sugere, é uma linguagem de script, onde a programação é realizada de uma forma mais leve e seu conteúdo é embutido diretamente no documento HTML. Vale a pena salientar que, apesar da semelhança de código, o JavaScript não é equivalente à linguagem de programação Java, que por sua vez é mais complexa e robusta.

Um recurso importante originado na linguagem JavaScript que foi mencionado nos artigos anteriores deste minicurso é o JSON (JavaScript Object Notation). Ele é um formato padronizado para a transferência de informações. Tem como características ser leve, fácil de interpretar e suportado por diversas plataformas, lembrando que este recurso é empregado neste projeto.

Contudo, outro recurso extremamente importante presente nesta linguagem é o AJAX (Asynchronous JavaScript And XML). O AJAX é uma tecnologia baseada em JavaScript, CSS e XML muito utilizada pelos Webmasters no ambiente da Internet para possibilitar a requisição assíncrona das informações através da rede de uma forma rápida, ou em outras palavras, uma página HTML não necessita ser recarregada novamente para receber um novo conteúdo, sendo assim, o desenvolvedor determina qual área da página deve ser alterada para apresentar tal informação, diminuindo assim a transferência de dados e evidentemente o tempo. O AJAX, por sua vez, engloba outras diversas tecnologias importantes para sua aplicação. A **Tabela 3** apresenta os recursos envolvidos na implementação do AJAX.

Entretanto, além do AJAX também existem outras tecnologias semelhantes, onde o formato dos dados transmitidos é diferente do XML, como é o caso dos acrônimos AJAT e AJAJ. O AJAT (Asynchronous Javascript And Text) é utilizado para definir uma transferência de informações baseadas em texto e o AJAJ (Asynchronous Javascript And JSON), por sua vez, é aplicado quando a transmissão é baseada na notação padrão JSON.

Devido a sua grande flexibilidade, a linguagem JavaScript conta com uma infinidade de Frameworks e Plug-ins que podem ser empregados no desenvolvimento de um documento HTML. Dentre eles, um dos principais e mais populares da atualidade, além de ser gratuito, é a biblioteca jQuery, concebida por John Resig. Ela é basicamente um Framework com uma diversidade de funcionalidades que simplificam e agilizam o desenvolvimento com JavaScript, oferecendo recursos como animações, manipulação de eventos, interações, AJAX, dentre outros. Para utilizar a jQuery, basta obter o arquivo de script (jquery.js) diretamente do Website do projeto e incluir uma chamada ao mesmo na seção <head> do documento HTML em questão. No caso do RadPHP XE 2, não há a necessidade da adição deste recurso pois a ferramenta já incorpora o mesmo em um projeto de aplicação móvel.

O Framework jQuery possui funções próprias e uma maneira particular de desenvolvimento. Também é importante salientar que existem diversos Plug-ins e outros Frameworks que são baseados no jQuery, que por sua vez, podem ser utilizados em conjunto com a mesma nos projetos, como é o caso do jQueryMobile (jQM). Presente também no RadPHP XE 2, o Framework jQueryMobile é constituído por diversas funções e estilos visuais para os elementos HTML (temas), tendo como alvo o ambiente móvel, permitindo por exemplo, a aplicação de efeitos de transição para as páginas da WebApp de uma forma bem simples.

Nota do DevMan

O XML é uma linguagem de marcação padronizada baseada em tags, muito popular na Internet que é extensível, ou em outras palavras, o desenvolvedor pode criar e utilizar quantas tags julgar necessário para o documento. Este formato de arquivo também é baseado em texto e é muito utilizado para o armazenamento de informações.

Nota do DevMan

A ECMA (European Computer Manufacturers Association) é uma associação que surgiu no século XX com o intuito de padronizar Sistemas de Informação. Uma das suas padronizações foi denominada ECMA-262, utilizada para definir a linguagem de Script ECMAScript, que por sua vez é uma linguagem para a manipulação de objetos e realização de cálculos que tem como plataforma alvo a Web, sendo semelhante a linguagem Java.

PhoneGap

A PhoneGap é uma API (Application Programming Interface) Open Source concebida pela Nitobi, atualmente pertencente à Adobe, na qual utiliza grande parte destes recursos empregados na Web para permitir que os desenvolvedores elaborem aplicações que podem ser facilmente exportadas para vários Sistemas Operacionais da atualidade e, principalmente, para a plataforma móvel, apresentando a ideia de escrever o código fonte do software apenas uma vez para ser empregado em

Recurso	Descrição
XMLHTTPRequest	Utilizado para a transferência das informações entre as aplicações cliente e Servidora de uma forma assíncrona.
JavaScript	A base para o funcionamento do mesmo, permitindo a manipulação desejada da informação, e consequentemente, sua visualização em determinada área da página.
DOM	Para permitir a interação com a informação.
CSS	Utilizado para definir os estilos para a informação.
XML	Padrão adotado para a troca dos dados entre as entidades envolvidas.

Tabela 3. Principais tecnologias presentes no AJAX

diversos ambientes, sendo capaz inclusive de interagir com funcionalidades nativas de Smartphones, tais como acelerômetro, câmera, entre outros recursos, além de ser integrada ou completada por outros Plug-ins. A PhoneGap também pode ser facilmente empregada em uma ferramenta de desenvolvimento, como é o caso do Eclipse, ou ser encontrada nativamente, como é o caso do RadPHP XE 2 e do Adobe DreamWeaver CS 6.

Um ponto extremamente importante que deve ser levado em consideração ao utilizar a PhoneGap em projetos mais complexos, como o desenvolvimento de uma aplicação cliente Data Snap em conjunto com o RadPHP XE 2 ou outras ferramentas, trata-se da necessidade de se definir em tempo de projeto qual será a tecnologia base empregada, ou em outras palavras, o RadPHP é focado no desenvolvimento com a linguagem de programação PHP, que por sua vez é uma linguagem Server-Side (Servidor), implicando que a aplicação, mesmo que desenvolvida para operar diretamente no Sistema Operacional alvo como o Android, necessitará efetuar requisições a um servidor PHP acessível para processar determinadas respostas que fazem uso desta linguagem, uma vez que este processo também ocorre no ambiente Web. Entretanto, o RadPHP também suporta o desenvolvimento HTML e JavaScript, sendo assim, também existe a possibilidade de desenvolver uma aplicação através desta ferramenta sem que haja a necessidade de consumir um servidor PHP, simplesmente utilizando os recursos do AJAX apresentados anteriormente, sendo esta última alternativa adotada para este projeto. Sendo assim, os recursos da linguagem PHP e da ferramenta RadPHP são utilizados apenas em tempo de Design.

Tutorial

Início do projeto com o RadPHP

Para dar início ao desenvolvimento da aplicação móvel com o RadPHP XE 2, basta acessar o menu Arquivo>Novo>Outro> Projetos RadPHP e selecionar a opção Mobile Application. Este projeto foi gravado com o nome de VendasMobile e o formulário principal como UPrincipal.php. Como a biblioteca jQuery será adotada como o motor principal de toda aplicação e não existem chamadas ao PHP em tempo de execução, todas as telas que serão exibidas foram criadas em apenas um formulário, sendo que cada uma foi atribuída a um painel. Sendo assim, foi inserido um objeto do tipo MPanel, presente na aba Mobile, (chamado de mpPrincipal) para que cada nova tela (painel) seja carregado nele.

O MPanel é semelhante ao TPanel do Delphi, um container visual que pode ser utilizado para armazenar e organizar os componentes. Em seguida, foram adicionados mais três MPanels para cada tela desta primeira etapa do projeto, sendo elas as telas de configuração da conexão (mpConf Conexao), menu principal (mpMenu) e cadastro de pessoas (mpCadPessoa). Posteriormente, foram adicionados vários objetos do tipo MButton, MEdit e MLabel. Além destes, ainda foi utilizado um componente do tipo MobileTheme. Este componente é um recurso da jQueryMobile que permite alterar todo o tema dos objetos de um documento HTML. Para atribuir um tema aos objetos, basta acessar a propriedade Theme dos mesmos e definir esta propriedade para o MobileTheme1. A **Figura 1** apresenta todos os componentes inseridos e uma visão geral desta etapa da aplicação em tempo de Design.

Figura 1. Telas e componentes do formulário principal da aplicação

O primeiro painel é o mpPrincipal, definido de uma forma bem reduzida, com o intuito de apenas carregar os outros painéis, conforme mencionado anteriormente. O próximo painel (mpConf Conexao) é utilizado para a configuração da conexão com o Servidor de Aplicação DataSnap, permitindo ao usuário definir um endereço IP (Internet Protocol) e o número da porta de acesso. O painel central é utilizado como um menu para o acesso aos outros recursos e para concluir, o último possui os campos necessários para o cadastramento de uma nova pessoa.

Definida a parte visual da aplicação, a próxima etapa é iniciar a elaboração dos códigos. Neste momento é necessário inserir o código responsável por manipular todos os eventos através da jQuery, onde o mesmo pode ser adicionado na seção <head> do documento HTML. Entretanto, o RadPHP gera automaticamente toda codificação HTML, não permitindo por padrão que sejam atribuídos códigos em algumas seções do documento.

Diante deste fator, quando o projeto é uma aplicação móvel, um formulário descende da classe MPage, sendo assim, esta classe possui diversos métodos que podem ser facilmente sobrescritos ou estendidos para adicionar funcionalidades que não estão acessíveis através dos eventos padrões, de forma semelhante à cláusula **override** presente no Delphi. Como neste projeto o código deve ser inserido na seção <head> do documento, o método da classe MPage denominado dumpHeaderJavascript, deve ser estendido. Este método é responsável por gerar o código fonte Javascript que se destina à seção mencionada anteriormente. A **Listagem 1** demonstra como estender o método dumpHeaderJavascript e dar início ao desenvolvimento baseado na jQuery.

Listagem 1. Estendendo o método dumpHeaderJavascript() da classe MPage

```
01 function dumpHeaderJavascript() {
02    parent::dumpHeaderJavascript();
03    ?>
04    <script>
05    //begin js
06    var conexao = {
07       ip: '192.168.1.100',
08       porta: '8080',
09       path: '/DataSnap/Rest/TServerMethods1/',
10       servicos: new Array('ReverseString/',
11                   'GetPessoas/',
12                   'InserirPessoa/')
13    }
14    jQuery(document).ready(function($) {
15       $.ajaxSetup ({
16          cache: false
17       });
18    });
19    //end
20    </script>
21    <?php
22 }
```

Logo após a declaração da função, é utilizado um código para que o método atual da subclasse herde o conteúdo de acordo com o método especificado da superclasse através do comando parent::dumpHeaderJavascript. Posteriormente, são utilizados os caracteres "?>", cujo objetivo é finalizar a definição dos códigos em PHP. Logo em seguida, é adicionada a tag HTML <script>, que por sua vez determina o início de um bloco de códigos para uma linguagem de Script, que neste caso é o Javascript. Para que o RadPHP reconheça que este trecho é um script, também são adicionados os caracteres "//begin js" (linha 5). Definida a inicialização do script, é criado um objeto (linha 6) utilizando Object Literals, que por sua vez recebeu a denominação de conexão. O recurso de Object Literals pode ser aplicado para a definição de objetos e suas propriedades de uma forma bem prática, que podem conter valores pré-definidos. É possível notar que o objeto conexao possui a propriedade ip, destinada a armazenar o endereço do Servidor, a propriedade porta (número da porta de comunicação para o servidor DataSnap), path, que representa parte da URI (Uniform Resource Identifier) do Servidor, e o array servicos, que por sua vez contém uma lista de todos os serviços até o momento disponibilizados pelo Servidor de aplicação. Em seguida, a jQuery é utilizada para efetuar um teste com o intuito de verificar se o carregamento do documento HTML foi concluído para que os métodos possam ser utilizados, através da linha jQuery(document).ready (linha 14), criando também uma nova função (function($)). Também é definido o comportamento padrão para todas as requisições AJAX com a jQuery, através da utilização do comando ajaxSetup, onde neste exemplo o armazenamento em cache é desabilitado. Para concluir, são finalizados os blocos definidos anteriormente, <script> e begin js, com a utilização de //end e </script>, novamente é efetuada a inicialização para a linguagem PHP, através dos caracteres <?php>.

Definida a base para o funcionamento da jQuery, será criada uma nova função cujo objetivo é "trocar" um painel pelo outro para a exibição. Vale à pena ressaltar também que a jQM (jQuery Mobile) conta com um recurso que permite a definição de várias páginas em apenas uma, com a utilização do elemento HTML e o atributo "data-role", onde posteriormente se emprega a função $.mobile.changePage(). Com algumas alterações, este recurso também pode ser empregado, mas neste exemplo foi definida uma forma manual para efetuar este processo. Também é importante evidenciar que todos os códigos seguintes devem ser inseridos no bloco que contém a declaração base deste exemplo, ou seja, o bloco de jQuery(document).ready(function($) {...}). A **Listagem 2** apresenta o método Mudar para alterar entre um painel e outro, adotado para esta WebApp.

Listagem 2. Método Mudar()

```
function Mudar(divOcultar, divMostrar) {
   divOcultar.hide();
   divMostrar.appendTo($('#mpPrincipal'));
   divMostrar.fadeIn(1000);
}
```

Note que o método recebe dois parâmetros, sendo o primeiro o painel que será escondido e o segundo, o que será exibido. No início da função, o objeto presente em divOcultar é ocultado através do método hide presente na jQuery. O objeto divMostrar é inserido no painel (mpPrincipal) através do método appentTo e exibido em seguida utilizando um efeito de transparência gradual (fade-in) através do método fadeIn, tendo como parâmetro o tempo de duração. No trecho de código, para acessar um elemento do documento HTML de uma forma bem simples através da jQuery, basta utilizar o comando $.('[id do elemento]'), que por sua vez tem seu funcionamento de forma semelhante ao método getElementById() da linguagem Javascript.

Concluída a função base, o próximo passo é inserir o código de inicialização para o documento HTML, que por sua vez irá ocultar todos os painéis presentes e exibir apenas o primeiro. A **Listagem 3** demonstra o código de inicialização utilizado.

Listagem 3. Código de inicialização

```
$('div:jqmData(role="content")').css('display','none');
$('#mpPrincipal').show();
$('#mpConfConexao').appendTo($('#mpPrincipal'));
$('#mpConfConexao').fadeIn(1000);
```

No RadPHP quando um painel é inserido, ele recebe por padrão o atributo data-role="content". Sendo assim, para ocultar todos os painéis rapidamente, basta utilizar as informações da jQM, neste caso contidas em jqmData, para todos que possuem este atributo e utilizar o método css da jQuery para alterar o estilo visual dos mesmos utilizando o recurso de CSS. Neste exemplo a propriedade display é definida como none para esconder os elementos. Outra forma de efetuar esta ação com a jQuery sem utilizar a jQM

seria através do comando $('div[data-role="content"]'), seguido pelo css e seus parâmetros. Posteriormente, o painel principal é exibido através do método show. O painel referente à conexão (mpConfConexao) é inserido no principal através do appentTo, onde posteriormente é aplicado o efeito de fade-In com sua efetiva duração.

O próximo passo é inserir os códigos responsáveis pela manipulação dos eventos de clique dos botões. A **Listagem 4** demonstra o código que é utilizado este fim.

O primeiro evento (linha 1) é destinado ao clique do botão Confirmar (btnConfirmar). Neste momento, é efetuada uma

Listagem 4. Código para os eventos de clique dos botões

```
01 $('#btnConfirmar').click(function() {
02    if (($('#edtIP').val() == '') || ($('#edtPorta').val() == '')){
03       alert('Dados incompletos!');
04    } else {
05       conexao.ip = $('#edtIP').val();
06       conexao.porta = $('#edtPorta').val();
07       Mudar($('#mpConfConexao'),$('#mpMenu'));
08    }
09    return false;
10 });
11 $('#btnConfConexao').click(function() {
12    Mudar($('#mpMenu'),$('#mpConfConexao'));
13    return false;
14 });
15 $('#btnCadPessoa').click(function() {
16    Mudar($('#mpMenu'),$('#mpCadPessoa'));
17    return false;
18 });
19 $('#btnCancelarPessoa').click(function() {
20    Mudar($('#mpCadPessoa'),$('#mpMenu'));
21    return false;
22 });
23 $('#btnGravarPessoa').click(function() {
24    var Nome = $('#edtNome').val() + '/';
25    var Endereco = $('#edtEndereco').val() + '/';
26    var Cidade = $('#edtCidade').val() + '/';
27    var Telefone = $('#edtTelefone').val() + '/';
28    var Uri = 'http://' + conexao.ip + ':' + conexao.porta +
29          conexao.path + conexao.servicos[2] +
30          Nome + Endereco + Cidade + Telefone;
31    $.ajax({
32       type: 'GET',
33       url: Uri,
34       contentType: 'application/json; charset=utf-8',
35       dataType: 'json',
36       success: function (data) {
37          $.each(data, function (i, Item) {
38             alert(Item.toString());
39          });
40       },
41       error: function () {
42          alert('Houve um erro!');
43       }
44    });
45    Mudar($('#mpCadPessoa'),$('#mpMenu'));
46    return false;
47 });
```


CURSOS ONLINE

A Revista Clube Delphi oferece aos seus assinantes uma série de Cursos Online de alto padrão de qualidade.

Clube Delphi

CONHEÇA ALGUNS DOS CURSOS:

- Rad Studio 2007-DBX4 e Firebird 2.0
  Aplicação comercial de vendas(básico) do Inicio ao Fim
- Dominando a Orientação a Objetos e Componentes em Delphi (Avançado)
- Novidades no Delphi 2010
- Curso Loja Virtual em Delphi Prism
- Aplicação Financeira com Delphi 2009 e MySQL

Para mais informações :
www.devmedia.com.br/cursos/delphi
(21) 3382-5038

DevMedia

validação para verificar se os campos de preenchimento (Edit) foram preenchidos e de acordo com a informação fornecida, o objeto conexao recebe os dados, que por sua vez são utilizados para a conexão com o Servidor de aplicação DataSnap. A função val da jQuery (linha 2) permite que um valor seja obtido ou atribuído a um objeto de uma forma simples e rápida. Para concluir, é efetuada a troca dos painéis e utilizado o comando return false, fazendo com que o comportamento padrão que seria adotado para o objeto em questão não seja executado, o que ocasionaria um recarregamento do documento HTML neste caso. Nos eventos de clique dos próximos três botões, é apenas utilizada a função Mudar em conjunto com o return false. Posteriormente, é atribuído o código para o evento de clique do botão de gravação para uma Pessoa (btnGravarPessoa). Como pôde ser observado, são criadas variáveis e os valores dos campos (Edit) são atribuídos as mesmas utilizando a função val (linhas 24 à 27). A seguir, é criada uma variável com a denominação de Uri, a fim de armazenar o endereço completo para o acesso ao servidor de aplicação DataSnap. Esta variável também recebe os quatro parâmetros necessários para o método disponibilizado pelo servidor, InserirPessoa. Em seguida, é utilizada a função ajax da jQuery. Esta função, assim como seu nome sugere, permite a utilização do AJAX (linha 31), possuindo diversas propriedades que podem ser definidas para que a solicitação desejada seja efetuada, bem como a manipulação das informações. Nesta função, a propriedade type é utilizada para definir o tipo da solicitação, que neste caso é GET, onde os valores são enviados diretamente com a própria URI. A propriedade url, empregada para indicar o endereço do servidor para a requisição, é definida com o valor da variável Uri. Em contentType e dataType é definido o tipo de retorno da solicitação, neste caso o formato adotado é o padrão JSON. Em success, deve ser definida a função que será executada caso o processo ocorra de forma satisfatória, que neste contexto é a exibição de uma mensagem através do comando alert e caso contrário, é executado o código presente em error. Para concluir, é alterado o painel para o menu principal e efetuado o return false.

Como pôde ser notado, a biblioteca jQuery é um Framework extremamente importante e útil para a manipulação dos eventos dos objetos do documento HTML de uma forma mais simples com o Javascript, aumentando a produtividade e facilitando o desenvolvimento da aplicação móvel que posteriormente será exportada com base na API PhoneGap.

Conclusão

A PhoneGap trouxe diversas vantagens para o desenvolvimento de aplicações para vários Sistemas Operacionais móveis com a utilização dos diversos padrões da Web, como o HTML e o Javascript, para a elaboração do Software, dispensando que os desenvolvedores habituados com estas linguagens tenham que conhecer as particularidades de cada plataforma móvel para só então poderem escrever seus projetos para as mesmas.

Como a linguagem Javascript é bem flexível, o desenvolvedor pode elaborar suas WebApps com base em Frameworks ou bibliotecas, como é o caso da jQuery e da jQueryMobile, com o intuito de simplificar e agilizar todo o trabalho, uma vez que diversas funções e métodos já estão prontos.

Giuliano Scombatti Pinto
giuliano@sygnux.com.br
É Analista Desenvolvedor da Sygnux Software (www.sygnux.com.br) localizada em Monte Alto/SP, atuando também com desenvolvimento Web e Android. Formado pela Fatec/TQ, começou a desenvolver com Delphi em 2003. Atualmente trabalha com Delphi, Java, PHP e Flash/Flex. Professor de Informática de Projetos Sociais da Prefeitura Municipal de Monte Alto/SP.

Site oficial e manual do PHP
http://www.php.net/manual/pt_BR/

Maiores informações sobre JSON
http://www.json.org

Site oficial da biblioteca jQuery
http://www.jquery.com/

Maiores informações sobre o framework jQueryMobile
http://www.jquerymobile.com/

DevSpace do Autor
http://www.devmedia.com.br/space.asp?id=283274

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

DevMedia

DÊ UM SALTO EM CONHECIMENTO!

Acesse o maior portal para desenvolvedores da América Latina!

17 mil posts

280 mil cadastrados

5 milhões de page-views por mês

http://www.devmedia.com.br/

DevMedia

dbExpress de A a Z – Parte 2

DataSnap em ação

Este artigo faz parte de um curso

A tecnologia DataSnap contempla aplicações de comunicação cliente-servidor, onde o servidor pode ser desenvolvido em Delphi ou C++ Builder, mas com o diferencial sobre o cliente, onde este pode ser desenvolvido em qualquer linguagem de programação que suporte JSON. A transferência de dados pode ocorrer através dos protocolos de comunicação TCP/IP, HTTP (onde o HTTP também pode utilizar REST – **Nota DevMan 1**) e HTTPS. Tudo isso se dá através de componentes de comunicação contidos no Delphi. Todo o envio e recebimento de dados entre cliente-servidor é obtido através da troca de conteúdo via JSON (**Nota DevMan 2**), onde a resposta do servidor também pode ser enviada em formato HTML, utilizando a tecnologia WebBroker (**Nota DevMan3**).

Com isto, temos em mãos um Framework para trabalhar com a arquitetura multicamadas no Delphi de forma simples, fácil e eficaz. Através do DataSnap, antigamente conhecido como MIDAS, podemos integrar o Delphi a praticamente tudo.

Isso facilita para o desenvolvedor quando a necessidade é ter um único ponto de desenvolvimento cuja integração precisa ser feita com outras tecnologias ou até mesmo por questões de escalabilidade, gerando aplicativos menores e mais leves para a camada Cliente e deixando o processamento maior e mais pesado para a camada Servidora, tornando assim, um projeto que utiliza o Framework DataSnap capaz de ser mais rápido, seguro e de fácil atualização junto aos diversos Clientes que podem estar conectados com o Servidor.

Ao trabalhar com esta arquitetura temos vários tipos de clientes que podemos utilizar, podemos servir informações do Delphi para aplicativos desenvolvidos em .Net (ASP.Net, WPF, Windows Forms, Windows Phone e outros), Java, FireMonkey, PHP entre outras linguagens; além de também ser possível a integração com outras plataformas móveis, tais como o iOS (iPhone e iPad) e Android (Java).

Temos com isto uma expansão dos horizontes na programação Delphi, o desenvolvedor que antes se via restrito a um desenvolvimento Desktop, hoje pode ter seu aplicativo rodando simultaneamente em diversas plataformas, dispositivos e Browsers com um único ponto de desenvolvimento.

Resumo DevMan

De que se trata o artigo:

Instruir o leitor a trabalhar com as melhores práticas em relação ao acesso a dados dentro do Delphi, utilizando o Framework dbExpress 4.0 e trabalhando com conceitos propostos pela Embarcadero Brasil. Nesta segunda parte iremos tratar o desenvolvimento Multicamadas para acessar os dados. Utilizaremos da tecnologia denominada DataSnap para transportar dados, objetos e quaisquer tipo de informações através de vários Clientes e ou Servidores.

Em que situação o tema é útil:

Disponibilizar nossos dados e processamentos através de um servidor de aplicação. A aplicação desse conceito é útil para centralizar as regras de negócio em um único local, contudo seus dados podem ser disponibilizados de forma online, podendo ser acessados por qualquer outro Software ou dispositivo.

dbExpress de A a Z – Parte 2:

Nesta série de artigos iremos abordar a suíte de conectividade dbExpress que ao longo dos anos sofreu grandes transformações, tornando-se hoje um Framework indispensável para quem deseja criar aplicações que possuam multiconectividade e performance sem grandes esforços. Além das vantagens abordadas, o dbExpress possui uma integração simples e ágil com a tecnologia DataSnap, que permite a criação de aplicações multicamadas, o que hoje realizam a integração entre aplicações heterogêneas, como WebSites e dispositivos móveis. Abordaremos os princípios do dbExpress, comentando seu funcionamento de uma maneira completa e abrangente.

Nota do DevMan 1

REST acrônimo de "Representation State Transfer" ou transferência de estado representacional é uma técnica de Engenharia de Software para sistemas hipermídia. O REST propõe o uso do protocolo HTTP para fornecer serviços onde os mesmos podem ser disponibilizados através de URL's utilizando as interfaces já conhecidas da Web (GET, POST, PUT e DELETE). Também podemos utilizar com o REST parâmetros de URL para passar dados para o servidor.

Nota do DevMan 2

JSON acrônimo de "JavaScript Object Notation" é um formato leve para a troca de dados computacionais, mais leve que o XML e suportado por diversas linguagens. JSON é um subconjunto da notação JavaScript, mas seu uso não requer JavaScript exclusivamente. O JSON propõe que seu uso seja mais rápido que outros formatos e que seu entendimento seja mais fácil a olho nu.

Nota do DevMan 3

O WebBroker é uma tecnologia que surgiu no Delphi 3, onde apresentou uma nova maneira de se desenvolver utilizando Delphi porém, com foco na Web. Com ela era possível realizar integrações e criações de páginas utilizando o modelo CGI (dll).

Neste artigo iremos abordar o processo inicial de desenvolvimento com o DataSnap, visando os melhores padrões de desenvolvimento e trabalho com esta tecnologia. Veremos aqui como criar um aplicativo Cliente e um Servidor na plataforma Delphi, além disso, teremos uma visão sobre como fazer esta integração com outras plataformas para que nas próximas partes do artigo seja possível abranger uma vasta linha de desenvolvimento.

O dbExpress com o conjunto de classes que estão contidas na unit DbxCommon, além de possibilitar trabalhar com os diversos bancos que dados que temos suporte no Delphi, também é a base para o trabalho com a característica de aplicativos multicamadas. Através dessa estrutura é que iremos dispor dados de uma tabela, métodos de um servidor de aplicação e até mesmo transferir objetos ou listas de objetos para os Clientes que conectarem ao Servidor de aplicações.

Na **Figura 1** é demonstrado um exemplo da arquitetura multicamadas do DataSnap, onde é perfeitamente possível visualizar as múltiplas integrações entre tecnologias diferentes. Podemos visualizar diferentes clientes, que englobam as ferramentas da Embarcadero, representadas pelo RAD PHP, Prism XE2, C++, Delphi, além dos dispositivos. Toda esta integração é realizada através de servidores que disponibilizam métodos conforme a necessidade. A última camada é representada pela multi-conectividade e independência de bancos de dados, na qual entra o principal tema deste artigo, o dbExpress.

Tutorial

Criando o servidor de aplicações

Para que as informações possam ser acessadas é necessário antes de tudo configurar e tornar o servidor disponível, seja localmente ou através da Internet. O DataSnap é dotado de diversas possibilidades de criação de um servidor, dentre elas estão disponíveis: DataSnap REST Application, DataSnap Server e DataSnap WebBroker Application. Para a resolução deste artigo foi utilizado o DataSnap Server, utilizando VCL Forms Application e os protocolos de comunicação TCP/IP e HTTP, cada um configurado na sua respectiva porta padrão e/ou porta disponível em seu computador. Para se criar este servidor recomenda-se a utilização do assistente Wizard que está disponível no Delphi através do menu File>New>Other>DataSnap Server.

A arquitetura de um servidor é simples. Basicamente é composta pela Interface de gerenciamento do mesmo, no caso representado pelo formulário disponível na criação da aplicação VCL Forms (**Figura 2**), a classes que serão servidoras de informações e a classe Container que abriga o servidor e seus respectivos componentes responsáveis pelos protocolos de comunicação e filtros de compressão, criptografia e etc.

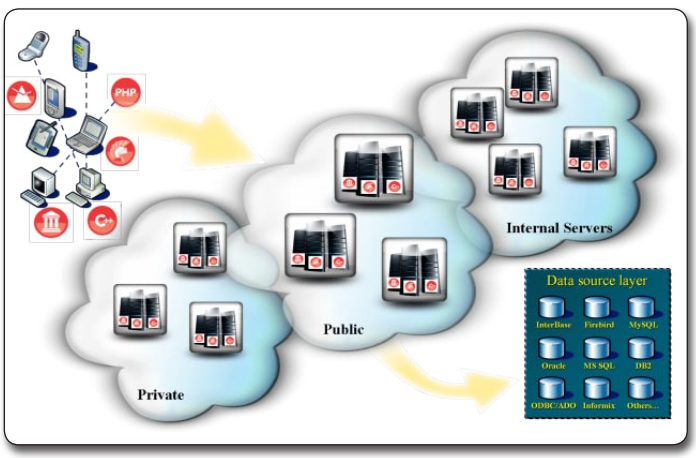


Figura 1. Arquitetura DataSnap

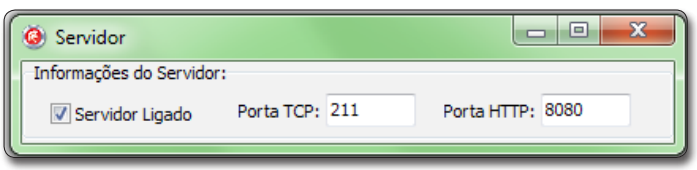


Figura 2. Layout da tela de gerenciamento do Servidor DataSnap

Como sugere a Orientação a Objetos, é importante manter as classes bem separadas de forma que isto irá facilitar o processo de manutenção, as classes servidoras de métodos (Server MethodsClass) são responsáveis por abrigar toda a informação que será transportada pelo servidor, logo, sugere-se que para cada tabela de banco seja criada uma classe de Server Methods (**Nota DevMan 4**).

Com isto temos um dinamismo na manutenção além de trabalhar de forma mais coerente os nossos arquivos. A Arquitetura atual do DataSnap não possui um limite estabelecido para o número de classes ServerMethos que podem estar contidas dentro de um Servidor.

As classes de ServerMethos devem herdar no mínimo de TPersistent, porém, o sugerido pelo é que herdem de TComponent, TDataModule ou TDSServerModule. Sendo mais indicada a última opção por se tratar de uma classe que é preparada exclusivamente para este tipo de processo. Isto se dá ao fato que o TDSServerModule foi concebido com a diretiva {$MethodInfo ON} que quando ativa, permite que informações sobre a classe sejam obtidas em tempo de execução através da RTTI, o que facilita o acesso aos métodos, propriedades, campos e dependências das classes.

Para exemplificarmos a vasta capacidade de um servidor Data Snap, para este exemplo nossa classe de ServerMethods1 chamada de TSrvMthdsPrincipal terá três métodos simples e um SQLDataSet, onde todas as informações serão enviadas para o Cliente de forma que o mesmo possa trabalhar com ClientDataSets ou até mesmo com classes Proxy para a conexão e manipulação dos dados.

Nota do DevMan 4

O princípio da responsabilidade única (SRP) recomenda que uma classe só deva ter um motivo para sofrer alguma alteração. Sendo assim, podemos resumir que uma classe qualquer em seu sistema deva ter uma tarefa única e específica. Isso, contudo não quer dizer que se tenha apenas um único método, mas que todos os métodos criados trabalhem juntos em um único objetivo, atender a responsabilidade da classe. Outros autores também chamam este princípio de coesão.

Para entendermos seu significado vamos imaginar um cenário que todos conhecem. Digamos que você possui uma classe que represente uma Nota Fiscal, a classe TNotaFiscal. E você a codificou de tal forma que ela mesma, além de conter seus itens, seus cálculos, também sabe como se salvar e se recuperar do banco de dados e até mesmo sabe como se exportar nos mais diversos formatos. Se pararmos para pensar, você acha que é responsabilidade da TNotaFiscal saber tudo isso? Apesar da forma que foi construída, temos essa classe lidando com mais de uma situação de responsabilidades diferentes. Se amanhã a forma de exportação é alterada, é preciso corrigir a classe. Se também algo muda na regra de negócio referente a uma nota fiscal, essa classe também precisa ser corrigida, ou seja, temos aqui mais de um motivo para essa classe ser alterada, e isso não é bom. Não é bom porque cada responsabilidade é um arco de mudanças, que muito provavelmente estão entrelaçadas. Estamos falando aqui de responsabilidade, mas o que é isso para uma classe? Ela representa um motivo de mudança. Nem sempre é fácil identificar responsabilidades a mais em uma classe, isso porque estamos acostumados a ver como um todo. Por exemplo, veja a classe TCliente a seguir.

```
TCliente = class
 private
  FDataNascimento: TDateTime;
  FId: integer;
  FNome: string;
  procedure SetDataNascimento(const Value: TDateTime);
  procedure SetId(const Value: integer);
  procedure SetNome(const Value: string);
 public
  property Id: integer read FId write SetId;
  property Nome: string read FNome write SetNome;
  property DataNascimento: TDateTime read FDataNascimento write SetDataNascimento;
  procedure Salvar(cliente: TCliente);
  procedure Excluir(cliente: TCliente);
  function Localizar(nomeCliente: string) : TList<TCliente>;
end;
```

Esse é um design muito comum. Observe e veja que, além de ter a responsabilidade de manter os dados dos clientes, é responsabilidade da classe TCliente se salvar, excluir e se localizar em algum banco de dados. Estamos acoplando, adicionando responsabilidades a mais, que poderiam inviabilizar a reutilização de TCliente. No código seguinte temos uma outra proposta, que respeita o princípio SRP.

```
TClienteSRP = class;

IDaoClienteSRP = interface
 ['{FC3651D9-3B1F-459F-B1C4-6EAA4616E0BC}']
  procedure Salvar(cliente: TClienteSRP);
  procedure Excluir(cliente: TClienteSRP);
  function Localizar(nomeCliente: string): TList<TClienteSRP>;
end;

TClienteSRP = class
 private
  FDataNascimento: TDateTime;
  FId: integer;
  FDao: IDaoClienteSRP;
  FNome: string;
  procedure SetDao(const Value: IDaoClienteSRP);
  procedure SetDataNascimento(const Value: TDateTime);
  procedure SetId(const Value: integer);
  procedure SetNome(const Value: string);
 published
 public
  property Id: integer read FId write SetId;
  property Nome: string read FNome write SetNome;
  property DataNascimento: TDateTime read FDataNascimento write SetDataNascimento;
  property Dao: IDaoClienteSRP read FDao write SetDao;
  constructor Create(dao: IDaoClienteSRP); reintroduce;
end;
```

Este design agora sim é superior ao anterior, porque separamos as funções de acesso a banco de dados em outra classe, representada pela interface IDaoClienteSRP. Mesmo estando em uma classe separada, nós ligamos a classe TCliente à IDaoClienteSRP através uma propriedade chamada Dao, embora fazer isso não seja o ideal, em alguns casos pode ser necessário. Este princípio é o mais simples, porém é o mais difícil de ser feito certo, pois exige boa interpretação, identificação e bom senso do desenvolvedor.

Para elaborarmos o nosso servidor de aplicação iremos implementar alguns métodos que estarão disponíveis para a aplicação Cliente. Para todos estes métodos a declaração deve ser realizada no escopo public. A **Listagem 1** ilustra um simples método denominado ObterDataHoraServidor, que conforme descrito, é autoexplicativo.

O segundo método a ser criado será o ObterTotalClientes (**Listagem 2**). Este método irá utilizar um SQLDataSet para conectar ao banco de dados e exibir o total dos clientes, cujo SQL para obter esta informação encontra-se na **Listagem 3**.

Listagem 1. Método ObterDataHoraServidor

```
function TSrvMthdsPrincipal.ObterDataHoraServidor: TDateTime;
begin
  Result := Now;
end;
```

Listagem 2. Metodo ObterTotalClientes

```
function TSrvMthdsPrincipal.ObterTotalClientes: Integer;
begin
  SqldtstTotalClientes.Open;
  try
    Result  := SqldtstTotalClientesTotalClientes.AsInteger;
  finally
    SqldtstTotalClientes.Close;
  end;
end;
```

Listagem 3. SQL para obter o total de clientes

```
Select
  count(*) as TotalClientes
from
  Clientes
```

Para o terceiro método será utilizada uma nova classe (**Listagem 4**) TPedido, a qual possui propriedades referentes aos campos do banco de dados. Foi criado o método ObterPedido que instancia um novo objeto, preenche as propriedades automaticamente e retorna o próprio objeto.

Na **Listagem 5** foi criado um método que retornará no momento da chamada o objeto Pedido criado anteriormente na **Listagem 4**. Perceba que o resultado é o método ObterPedido, que retorna o próprio pedido já preenchido.

Após as implementações nosso ServerMethod já está pronto para ser transportado através do ServerContainer, algo importante a se frisar é a questão do nome da classe pois, será através desta que tanto o ServerContainer quanto os clientes conectados ao servidor irão identificar o local de onde buscar as informações.

Na classe de ServerContainer teremos os componentes responsáveis por tornar o aplicativo um devido servidor DataSnap, aqui possuímos os seguintes componentes:

• **TDSServer**: Este componente é o servidor DataSnap em questão, por ele são efetuadas as rotinas de Start e Stop, que respectivamente iniciam ou param o servidor, além dos eventos que são disparados quando um cliente conecta ou desconecta do mesmo.

Listagem 4. Classe TPedido

```
type
  TPedido = class
  private
    FPED_CODIGO: Integer;
    FCLI_CODIGO: Integer;
    FPED_DATA: TDateTime;
    FPED_DATA_ENTREGA: TDateTime;
    FTotalItens: Integer;
  public
    property PED_CODIGO: Integer read FPED_CODIGO write FPED_CODIGO;
    property CLI_CODIGO: Integer read FCLI_CODIGO write FCLI_CODIGO;
    property PED_DATA: TDateTime read FPED_DATA   write FPED_DATA;
    property PED_DATA_ENTREGA : TDateTime read FPED_DATA_ENTREGA
    write FPED_DATA_ENTREGA;
    property TotalItens: Integer   read FTotalItens write FTotalItens;
    function ObterPedido(const Codigo : Integer) : TPedido;
  end;

implementation

uses
  System.SysUtils, System.DateUtils;

function TPedido.ObterPedido(const Codigo: Integer): TPedido;
var Pedido : TPedido;
begin
  Pedido  := TPedido.Create;
  Pedido.PED_CODIGO:= Codigo;
  Pedido.CLI_CODIGO:= 1;
  Pedido.PED_DATA:= Now;
  Pedido.PED_DATA_ENTREGA:= IncDay(Now, 5);
  Result:= Pedido;
end;
```

Listagem 5. Método ConsultarPedido

```
function TSrvMthdsPrincipal.ConsultarPedido(
  const CodigoPedido: Integer): TPedido;
var Pedido : TPedido;
begin
  Pedido  := TPedido.Create;
  try
    Result  := Pedido.ObterPedido(CodigoPedido);
  finally
    Pedido.Free;
  end;
end;
```

É o principal responsável por gerenciar a vida útil das classes que são responsáveis por transportar as informações contidas na camada servidora.

• **TDSTCPServerTransport**: Componente responsável pela configuração e uso do protocolo TCP/IP dentro do servidor.

• **TDSHTTPService:** Componente responsável pela configuração e uso do protocolo HTTP dentro do servidor. Também responsável pela capacidade de permitir que um cliente invoque métodos utilizando REST, além de poder servir de gestor de autenticações, implementadas pelo componente TDS AuthenticationManager, caso se faça necessário este tipo de implementação. A classe TDSAuthenticationManager conforme o próprio nome sugere, realiza o gerenciamento de conexões

com o servidor de aplicações. Através dela é possível adicionar mais segurança em suas aplicações criando formas de autenticação (uma espécie de senha para a conexão com o servidor), muito semelhante ao que é feito no processo de configuração de conexão com o banco de dados hoje.

• **TDSServerClass** - Componente utilizado para especificar uma classe servidora de métodos que poderá ser chamada dinamicamente por um cliente conectado ao Servidor.

Para que o DsServerClass possa transportar a classe correta para os clientes que efetuarem a chamada, é necessário implementar o evento OnGetClass retornando a própria classe para a variável PersistentClass conforme a **Listagem 6**.

Finalizando a implementação da classe **SrvCntrPrincipal** ainda é necessário efetuar a implementação de dois métodos que serão utilizados na manutenção do servidor. O primeiro (LigarDesligarServidor) tem apenas o objetivo de iniciar ou parar o servidor de aplicação, já o segundo retorna as portas que estão sendo utilizadas pelos componentes de protocolos TCP/IP e HTTP respectivamente (**Listagem 7**).

Listagem 6. Evento onGetClass do componente DSServer

```
uses Winapi.Windows, UntSrvMthdsPrincipal;

procedure TSrvCntrPrincipal.DSServerClassPrincipalGetClass(
 DSServerClass: TDSServerClass; var PersistentClass: TPersistentClass);
begin
 PersistentClass := UntSrvMthdsPrincipal.TSrvMthdsPrincipal;
end;
```

Listagem 7. Métodos restantes na classe SrvCntrPrincipal

```
procedure TSrvCntrPrincipal.LigarDesligarServidor(const Ligado: Boolean;
 const PortaTCP, PortaHTTP: Integer);
begin
 if (Ligado) then
 begin
  DSServer.Stop;
  DSTCPServerTransport.Port := PortaTCP;
  DSHTTPService.HttpPort   := PortaHTTP;
  DSServer.Start;
 end else
  DSServer.Stop;
end;
```

A listagem a seguir ilustra o método ObterPortasServidor, na qual simplesmente realiza a configuração dos parâmetros referentes as portas do servidor de aplicação.

```
procedure TSrvCntrPrincipal.ObterPortasServidor(var Portas: array of Integer);
begin
 Portas[0] := DSTCPServerTransport.Port;
 Portas[1] := DSHTTPService.HttpPort;
end;
```

A implementação final do formulário (**Listagem 8**) principal fecha o processo de criação do servidor, onde temos a ação de desativar/ativar o servidor na seleção do checkbox ChckbxStatusServidor (**Figura 2**) ChckbxStatusServidor e no momento da exibição do formulário, no evento FormShow, recuperamos o status do servidor e as portas que estão em uso. Rodando o aplicativo por fora do Delphi, iremos construir um cliente para conexão.

A camada cliente

A camada cliente desenvolvida em Delphi é de uma implementação excepcionalmente simples, requerendo apenas o componente SQLConnection para servir como conexão com o DataSnap através do driver padrão contido no Delphi. O aplicativo Cliente (**Figura 3**) demonstra o transporte de dados através do protocolo JSON e aplicação nativa desenvolvida em Delphi em ambos os lados (Cliente-Servidor). Pode-se notar o processo de conversão ou (Parser) das classes contidas no servidor onde um objeto TPedido é convertido para JSON, fazendo o processo inverso na camada Cliente. Isto acontece de forma implícita, facilitando o processo de desenvolvimento e tornando a experiência mais confortável para o desenvolvedor.

Listagem 8. Implementações restantes do formulário principal

```
uses UntSrvCntrPrincipal;

//Evento OnClick do CheckBox
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
 SrvCntrPrincipal.LigarDesligarServidor(
 ChckbxStatusServidor.Checked,
 StrToInt(EdtPortaTCP.Text),
 StrToInt(EdtPortaHTTP.Text));
end;

//Evento OnShow do formulário
procedure TForm1.FormShow(Sender: TObject);
var Portas : Array[0..1] of Integer;
begin
 SrvCntrPrincipal.ObterPortasServidor(Portas);
 EdtPortaTCP.Text          := IntToStr(Portas[0]);
 EdtPortaHTTP.Text         := IntToStr(Portas[1]);
 ChckbxStatusServidor.Checked := True;
end;
```

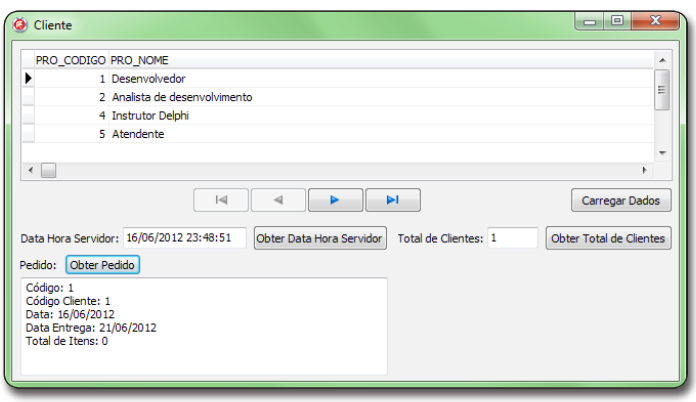


Figura 3. Interface do cliente DataSnap em funcionamento

O projeto cliente é uma nova aplicação do tipo VCL Forms Application, nessa aplicação foi adicionado um DataModule

que será utilizado para realizar a conexão com o servidor. Essa conexão se dá por um componente SQLConnection. Deste, mmodo, para o desenvolvimento desta camada devemos manter o aplicativo servidor em execução e ativo, de modo que o componente SQLConnection possa visualizar as classes disponíveis para o transporte. Isso é alcançado ao efetuar realizar oo processo de conexão. Para isso, basta selecionar o Data Explorer no menu de contexto View, opção Data Explorer. Selecione o driver DataSnap e clique de direita escolhendo a opção Add New Connection e informando o nome da conexão em seguida. Este nome é o identificador da conexão no momento da configuração do SQLConnection. A **Figura 4** mostra a configuração de alguns parâmetros importantes neste processo, dentre eles o Communication Protocol referente ao protocolo de comunicação que será utilizado para se conectar ao servidor. Podendo variar entre TCP/IP, HTTP e HTTPS. Nesta etapa também se deve configurar as propriedades o HostName e Port para finalizar o processo de

Uma vez conectado, para utilizar os dados que foram dispostos no servidor através da classe de ServerMethod obtém-se duas possibilidades, o consumo de informações vindas através de um TDataSet e um TDataSetProvider (do servidor) ou mesmo utilizando o componente TSQLServerMethod que garante acesso aos métodos disponíveis no servidor de aplicação. A conexão pode ser realizada através do componente TDSProviderConnection que funciona como uma espécie de comunicador com o servidor. Neste, o Parâmetro ServerClassName deve ser configurado com o nome da classe definida na variável PersistentClass (evento OnGetClass do DSServer), em outras palavras, deve-se informar o nome da classe pela qual a conexão será realizada. Essa foi a opção utilizada. (**Figura 5**)

Com o SQLConnection devidamente conectado ao servidor DataSnap e com o DSProviderConnection ligado à classe, é possível no componente ClientDataSet definir a propriedade RemoteServer que apontará para o DSProviderConnection. Logo que o componente DSProviderConnection está conectado à classe do servidor por meio da propriedade ClassName, a propriedade RemoteServer será capaz de enxergar todos os DataSetProviders disponíveis naquela determinada classe, permitindo assim a conexão com o ClientDataSet através da propriedade ProviderName, tornando assim o trabalho com o ClientDataSet de agora em diante idêntico ao que é na arquitetura convencional.

E para obtermos a classe de Proxy para o Delphi basta clicar de direita sobre o componente SQLConnection e selecionar a opção "Generate DataSnap Client Classes" que o mesmo se encarregará de buscar no servidor as classes disponíveis nos ServerMethods e trazer um cliente compatível com suas respectivas chamadas. As classes proxy são a grosso modo um "atalho" para os métodos do servidor de aplicação. Servem para intermediar o acesso a estes métodos, possibilitando a chamada de qualquer método do servidor a partir da criação da classe. A **Listagem 9** ilustra a classe Proxy automaticamente criada pelo SQLConnection.

Note que para realizar uma chamada de um método qualquer, basta utilizar a classe passando a conexão atual como parâmetro. A **Listagem 10** exibe como acessar o método ObterDataHoraServidor.

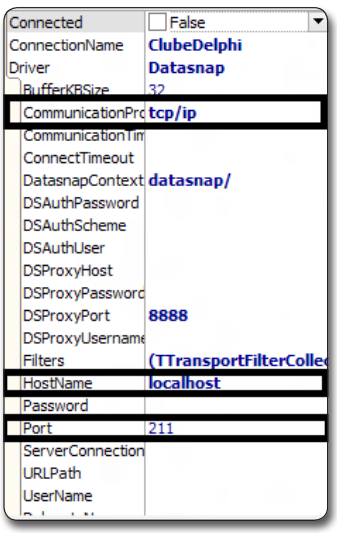


Figura 4. Parâmetros de configuração da conexão do SQLConnection

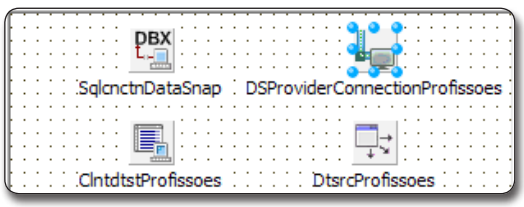


Figura 5. DataModule Cliente

Listagem 9. Classe Proxy gerada pelo SQLConnection

```
type
 TSrvMthdsPrincipalClient = class(TDSAdminClient)
 private
  FObterDataHoraServidorCommand: TDBXCommand;
  FObterTotalClientesCommand: TDBXCommand;
  FConsultarPedidoCommand: TDBXCommand;
 public
  constructor Create(ADBXConnection: TDBXConnection); overload;
  constructor Create(ADBXConnection: TDBXConnection;
   AInstanceOwner: Boolean); overload;
  destructor Destroy; override;
  function ObterDataHoraServidor: TDateTime;
  function ObterTotalClientes: Integer;
  function ConsultarPedido(CodigoPedido: Integer): TPedido;
 end;
```

Listagem 10. Exemplo de chamada de um método da Classe Proxy

```
var Classe :TSrvMthdsPrincipalClient;
begin
 Classe := TSrvMthdsPrincipalCliente.Create(SQLConnection.DBXConnection);
 try
  Classe. ObterDataHoraServidor;
 finally
  Classe.Free;
 end;
end;
```

Ainda tendo como base a classe Proxy gerada pelo Delphi, podemos visualizar o conceito de conversão implícita que é feita quando se transita objetos através do DataSnap. O método ConsultarPedido que retorna um objeto do tipo TPedido possui em sua implementação o código que destaca este processo de conversão, conforme a **Listagem 11,** na linha 7.

Listagem 11. Conversão de JSON para TPedido

```
01 ...
02  FUnMarshal:=
03  TDBXClientCommand(FConsultarPedidoCommand.Parameters[1].
04    ConnectionHandler).GetJSONUnMarshaler;
05  try
06    Result :=
07      TPedido(FUnMarshal.UnMarshal(FConsultarPedidoCommand.
08        Parameters[1].Value.GetJSONValue(True)));
09    if FInstanceOwner then
10      FConsultarPedidoCommand.FreeOnExecute(Result);
11  finally
12    FreeAndNil(FUnMarshal)
13  end
14 ...
```

Para a utilização dos demais recursos a implementação fica por conta do formulário principal, que utilizará código mesclando os recursos disponíveis através da classe Proxy e ClientDataSet. Para acessar as informações através do ClientDataSet, basta abri-lo para que as informações sejam exibidas como o de costume. Para os demais métodos, o consumo deve ser feito através da classe Proxy (**Nota DevMan 5)**.

Nota do DevMan 5

Classes proxy no são necessárias em um aplicativo cliente para invocar métodos do servidor, gerada automaticamente em clientes Delphi a partir de um TSQLConnection a classe proxy se conecta a uma execução do servidor DataSnap e proverá acesso a quaisquer função ou procedimento declarados na diretiva public dos ServerMethods contidos no servidor.

No código da **Listagem 12** a classe Proxy é criada por meio da variável Proxy do tipo TSrvMthdsPrincipalClient. Vale a pena salientar que ela é uma transposição daquilo que temos na aplicação servidora. Ainda é possível visualizar a criação do objeto Proxy passando como parâmetro a conexão com o servidor por meio do objeto DBXConnection. O resultado da chamada do método é automaticamente atribuído ao respectivo Edit. No código da **Listagem 13** é realizado um procedimento semelhante, porém, de forma que o resultado da consulta de total de pedidos inseridos no banco de dados seja exibido no Edit.

Diferente das listagens anteriores, aqui na **Listagem 14** é realizado um procedimento um pouco diferente. O objeto Proxy também é criado passando a conexão como parâmetro, porém, o objeto TPedido é retornado vindo do servidor por meio da chamada de ConsultarPedido. Note que os valores são obtidos através das próprias propriedades da classe TPedido. O resultado de tudo é armazenado no Memo como uma forma de detalhamento do pedido encontrado.

Listagem 12. Botão para obter data e hora do servidor

```
procedure TFrmPrincipalCliente.BttnObterDataHoraServidorClick
(Sender: TObject);
var Proxy: TSrvMthdsPrincipalClient;
begin
 Proxy = TSrvMthdsPrincipalClient.Create(

DtmdlCliente.SqlcnctnDataSnap.DBXConnection);
 try
  EdtDataHoraServidor.Text  :=
   FormatDateTime('dd/mm/yyyy hh:mm:ss',Proxy.ObterDataHoraServidor);
 finally
  Proxy.Free;
 end;
end;
```

Listagem 13. Obter total de pedidos

```
procedure TFrmPrincipalCliente.BttnObterTotalClientesClick(Sender: TObject);
var Proxy : TSrvMthdsPrincipalClient;
begin
 Proxy := TSrvMthdsPrincipalClient.Create(
  DtmdlCliente.SqlcnctnDataSnap.DBXConnection);
 try
  EdtTotalClientes.Text:= IntToStr(Proxy.ObterTotalClientes);
 finally
  Proxy.Free;
 end;
end;
```

Listagem 14. Obter pedido

```
procedure TFrmPrincipalCliente.BttnObterPedidoClick(Sender: TObject);
var Pedido: TPedido;
   Proxy: TSrvMthdsPrincipalClient;
begin
 Proxy:= TSrvMthdsPrincipalClient.Create(
  DtmdlCliente.SqlcnctnDataSnap.DBXConnection);
 try
  Pedido  := Proxy.ConsultarPedido(1);
  MmPedido.Lines.Add('Pedido:');
  MmPedido.Lines.Add('Código: '+ IntToStr(Pedido.PED_CODIGO));
  MmPedido.Lines.Add('Código Cliente: ' + IntToStr(Pedido.CLI_CODIGO));
  MmPedido.Lines.Add('Data: '+ DateToStr(Pedido.PED_DATA));
  MmPedido.Lines.Add('Data Entrega: '+ DateToStr(Pedido.PED_DATA_ENTREGA));
  MmPedido.Lines.Add('Total de Itens: ' + IntToStr(Pedido.TotalItens));
 finally
  Proxy.Free;
 end;
end;
```

Classes Proxy

A geração de classes Proxy é feita de forma automática através do SQLConnection como já foi mencionado, no entanto, o que é importante ressaltar destas classes é a forma como é feita a implementação da chamada de métodos, todos utilizam o dbExpress (Data.DBX Common, Data.DBXClient, Data.DBXDataSnap, entre outras).

Para gerar chamadas através da camada Servidora para a camada Cliente, é utilizada a classe TDBXCommand, que suporta

diversos processos de chamadas, desde acesso a dados em bancos diretamente do Delphi ao banco de destino, assim como chamadas de métodos a servidores DataSnap, através da configuração do CommandType. O importante deste processo é a característica de que todo comando é tratado da mesma maneira, seja este um comando para selecionar colunas em um banco de dados, uma chamada de Stored Procedure ou mesmo uma solicitação de um método remoto via DataSnap.

Outro ponto importante de se ressaltar é que ao criar um objeto do tipo que é gerado na classe Proxy, por padrão, o mesmo requer um DBXConnection que é a representação de uma conexão para o Framework dbExpress. Isto está associado à nova arquitetura que vem sendo trabalhada no dbExpress 4, onde fica disponível para o desenvolvedor a utilização de classes mais diretas para o acesso às informações.

O DBXConnection está acoplado ao TSQLConnection, facilitando o processo para o desenvolvimento, onde o desenvolvedor pode configurar de forma visual o SQLConnection e a cada vez que se fizer necessária a utilização do DBXConnection, o mesmo pode ser chamado através do SQLConnection.DBXConnection.

Classes Proxy em outras linguagens

Como foi dito anteriormente, o DataSnap é uma tecnologia capaz de integrar ambientes de desenvolvimento distintos. Isto se dá à capacidade de transportar objetos no formato JSON, o que nos dá o benefício de centralizar o desenvolvimento e tornar nossos softwares mais abrangentes.

Contudo, vale ressaltar esta utilização visando alguns pontos importantes de integração que o DataSnap nos proporciona ao centralizar todo o processo de desenvolvimento em uma única linguagem. Atualmente através do Delphi e FireMonkey podemos desenvolver aplicativos para dispositivos móveis (iOS) além de aplicativos multiplataforma (Windows, MacOS) contudo, o acesso a dados em diversos bancos e/ou ambientes, ainda possui uma grande dependência de plataforma, principalmente quando o assunto está relacionado à dispositivos móveis, por questões de capacidade de processamento e de acessibilidade.

Este problema é facilmente solucionado através da utilização do DataSnap, pois, com a utilização deste Framework como provedor de dados, os clientes que irão se conectar para obter algum tipo de informação não necessariamente deverão ter conhecimento da fonte de dados que está além da camada Servidora, bastando apenas uma integração entre clientes e servidores assim como foi visto neste artigo.

Conclusão

Neste artigo abordamos a parte de acesso a informações através do DataSap, desenvolvemos os primeiros passos para a criação e integração entre Cliente e Servidor, visualizamos os pontos importantes para o desenvolvimento dentro da camada Servidora. Através da estrutura aqui construída é possível se compartilhar informações através de qualquer linguagem de programação que suporte JSON, além de também ser a base para um desenvolvimento mais avançado, pois, quando trabalhamos com a perspectiva de servidor de aplicação, temos que pensar em metodologias de transporte eficazes para condensar melhor a informação que será transportada, de modo que a mesma seja levada de uma ponta à outra de forma segura, compacta e rápida.

Algo importante a se ressaltar no DataSnap é a simplicidade dos aplicativos Clientes, tornando-os cada vez mais independentes do local de funcionamento e de processamento, deixando por conta do servidor de aplicação a tarefa de cuidar de todo o trabalho árduo referente às grandes massas de dados.

Alexandre Borges
contato@all-s.info
Desenvolvedor, instrutor e consultor: Delphi, C#, iOS e Maker. Graduado em gestão de TI, certificado Delphi Developer, Delphi Master e Microsoft MCP. Instrutor Oficial da Embarcadero em Goiânia.

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Gerenciamento de Riscos

Boas práticas para gerenciar riscos em projetos de software

O MSF - Microsoft Solutions Framework - é um processo de desenvolvimento de software definido pela Microsoft. É um conjunto de boas práticas de engenharia de software, princípios e metodologias comprovadas destinadas a habilitar desenvolvedores a obter sucesso no ciclo de vida do desenvolvimento de aplicações orientadas a objeto, incluindo Delphi. O MSF provê orientações ágeis e adaptáveis, baseadas nas melhores práticas e experiências de grandes empresas (incluindo a própria Microsoft), para aumentar consideravelmente o sucesso na construção de aplicações na área de tecnologia da informação. Foca em soluções para o cliente baseadas em entregas rápidas, diminuindo o número de pessoas envolvidas no projeto, gerenciando e eliminando riscos constantemente, enquanto permite a entrega de resultados de alta qualidade (Microsoft, 2002).

Alguns dos seus princípios fundamentais são: promover uma comunicação aberta, trabalhar através de uma visão compartilhada, capacitar os membros da equipe, estabelecer responsabilidades de forma compartilhada, focar no valor do negócio, permanecer ágil e esperar por mudanças, investir em qualidade e aprender sobre todas as experiências.

O MSF possui cinco fases (**Figura 1**):

• **Envisioning** – Fase inicial que prepara um documento de visão e escopo, orientando todos os membros da equipe a terem uma visão única e clara dos objetivos do projeto. Identificação de riscos e preparação de planos de contingência são iniciados nesta fase;

• **Planning** – Nesta fase é criado o design da solução, a principal atividade é a definição de um planejamento que garanta o sucesso do projeto. O planejamento inclui custos, recursos e cronograma;

• **Developing** – Nesta etapa são criados os componentes da solução, baseados no planejamento;

• **Stabilizing** – Todos os componentes são testados para garantir que toda a solução funciona como planejado;

• **Deploying** – Após garantir, através de testes, que a solução é estável, ela pode então ser colocada em ambiente de produção.

Resumo DevMan

De que se trata o artigo:

Este artigo trata de uma importante disciplina no gerenciamento de projetos de software, o gerenciamento de riscos. Esta disciplina será abordada neste artigo dentro do contexto do MSF – Microsoft Solutions Framework, porém as técnicas aqui demonstradas também se aplicam a outros processos. Primeiramente é apresentada uma visão geral do MSF e suas fases. A seguir, a disciplina de gerenciamento de riscos é detalhada em suas seis fases: Identificar, Analisar, Planejar, Rastrear, Controlar e Aprender.

Em que situação o tema é útil:

Ao contrário do que muitos podem pensar, um risco não é um problema. Um risco é uma possibilidade, não uma certeza. Um risco é analisado, planejado, e a equipe se prepara cuidadosamente com planos de contingência caso ele se torne um evento (realmente ocorra). Prever riscos é muito útil, pois pode fazer com que a equipe trate o evento como uma oportunidade. Ou ainda, evitar que o projeto fracasse. Envolver o gerenciamento de riscos como uma disciplina constante é útil para reaproveitar a experiência em outros projetos.

Gerenciamento de Riscos:

A disciplina de gerenciamento de riscos não é exclusiva do MSF, muitas das boas práticas e experiências aqui apresentadas serão comuns a outros cenários. Aqui o termo disciplina significa que esta é uma atividade constante realizada durante todo o ciclo de vida do projeto. Riscos são constantemente monitorados, avaliados. Boas práticas ágeis como comunicação aberta, fluente e destemida, farão com que essa gerência se torne ainda melhor. Identificar riscos, controlá-los e aprender com eles são as melhores práticas propostas pelo MSF, que o leitor poderá incorporar em seu processo de desenvolvimento de aplicações Delphi.

Dentro do processo do MSF, existe uma disciplina realizada constantemente durante todo o ciclo de vida do projeto, chamada de *gerenciamento de riscos*. Esta disciplina não é um processo realizado à parte ou em paralelo aos processos vistos anteriormente, nem deve ser feito somente no início do projeto (uma tendência). É uma atividade contínua, evolutiva, com controle permanente, planejada juntamente com as demais atividades do projeto.

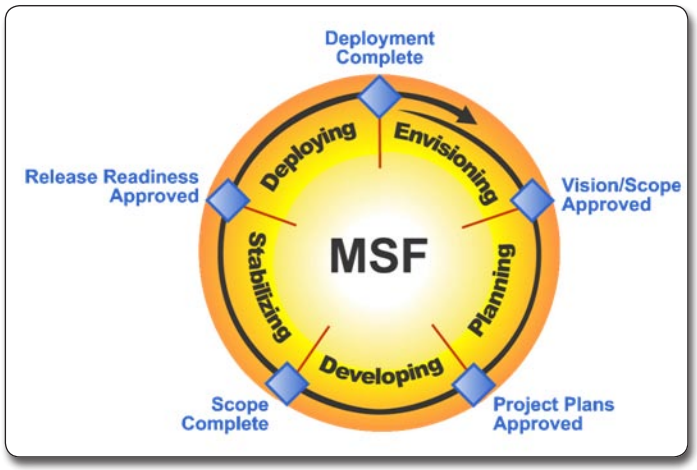


Figura 1. Visão geral do MSF (Fonte - Microsoft, 2002)

O foco deste artigo é detalhar todos os fundamentos desta importante atividade, o gerenciamento de riscos, dentro do contexto do processo do Microsoft Solutions Framework. Vale observar que as boas práticas aqui apresentadas também se aplicam a outras metodologias e processos, como RUP, XP e Scrum. Inicialmente serão apresentadas algumas boas práticas e boas experiências na aplicação desta disciplina. O artigo detalha também as fases envolvidas no processo de gerenciamento de riscos: identificar, analisar, planejar, rastrear, controlar e aprender. Mostra ainda quais são os documentos utilizados como entrada e saída em cada uma das fases.

Riscos e a importância do seu gerenciamento

Em gerenciamento de projetos, um aspecto importante a ser controlado são os riscos. Um risco pode ser descrito como um evento que pode ter um impacto positivo ou negativo no projeto. Riscos normalmente são associados a fatores como perda de controle, funcionalidade, qualidade ou atrasos. Um risco, no entanto, não é um problema. Um problema é um fator que ocorre de forma inesperada em um dado momento do projeto, quando ele já está em execução (*presente*), enquanto os riscos são cuidadosamente previstos no planejamento, ou seja, eles se referem a situações adversas no *futuro*.

O objetivo do gerenciamento de riscos é maximizar os impactos positivos (*oportunidades*) e minimizar os impactos negativos (*perdas*) relacionados aos riscos do projeto. Os projetos na área de tecnologia de informação (TI) possuem características próprias que tornam o gerenciamento de riscos um fato essencial para se obter sucesso, onde podemos citar: competividade empresarial, evolução de tecnologias, importância de fatores relacionados à segurança, mudança constante em requisitos e mudanças na equipe de desenvolvimento.

Boas práticas no gerenciamento de riscos

Uma boa prática no gerenciamento de riscos é ter um pensamento ágil, esperar por mudanças, pois elas certamente ocorrerão. O gerenciamento de riscos não deve se limitar a uma única fase do projeto, mas deve ser uma atividade constante durante todo o ciclo de vida da aplicação, pois o desenvolvimento do projeto de software é, segundo uma abordagem ágil, um processo contínuo, evolutivo e iterativo.

Para uma boa identificação de riscos, todo o time deve ter participação contínua na análise, incluindo stakeholders externos ao time.

Nota do DevMan

A palavra stakeholder significa "stake", interesse; e "holder", aquele que possui. Na prática são todos aqueles que influenciam uma empresa. São os interessados pelos projetos, gerenciamento, mercado e produtos de uma empresa. São os colaboradores, funcionários, clientes, consumidores, planejadores, acionistas, fornecedores, governo e demais instituições que direta ou indiretamente interfira nas atividades gerenciais e de resultado de uma organização

A comunicação contínua, aberta e fluente entre todos os envolvidos resultará em uma melhor identificação e análise de riscos, minimizando falhas. Além disso, a experiência adquirida pelos membros em projetos anteriores pode ajudar a identificar potenciais riscos em novos projetos. O objetivo principal é montar uma base de dados de conhecimento sobre riscos, constantemente alimentada, servindo com uma fonte valiosa de consulta. No time, ninguém em específico é o "proprietário" ou responsável pelo gerenciamento de riscos, cada um no time é responsável por ativamente participar deste processo.

Ao gerenciar riscos, é importante adotar uma abordagem proativa para identificar, analisar e planejar riscos focando os seguintes princípios:

- Antecipar problemas ao invés de apenas reagir a eles quando ocorrerem; Definir estratégias antecipadamente; Definir medidas de mitigação de risco e redução de impacto, ainda no planejamento, quando a equipe tem a capacidade de intervir; Usar medidas preventivas sempre que possível;
- Resolver a causa principal ao invés de lidar com os sintomas; Medidas corretivas focadas em causas podem ser de grande valia para empresa, pois normalmente podem ter um alcance maior do que o escopo de um projeto em particular;
- Ter a resolução planejada e pronta para problemas, antes que ocorram;
- Usar um mecanismo efetivo, conhecido, estruturado e repetitivo para a resolução de problemas.

O time deve sempre visualizar a identificação de riscos como algo positivo, fazendo isto sem medo, a fim de garantir a contribuição máxima no detalhamento destes e não gerar opiniões controversas. Uma percepção negativa de um risco pode causar relutância na comunicação. Em alguns ambientes, a pessoa que comunica um risco é tratada como um "causador de problemas" e a atenção é, a partir de então, dirigida a esta pessoa e não ao risco em si. Pessoas assim tendem a não expressar ideias sobre riscos para não confrontar os demais membros do time. (Microsoft, 2002)

É necessário que todos os membros do time tratem o gerenciamento de riscos como uma tarefa contínua e necessária, e não "chata" que deve ser realizada apenas no princípio do projeto. Mudanças contínuas no projeto fazem com que riscos também sejam continuamente monitorados e reavaliados para responder a mudanças, dentro da estrutura original do projeto, sem criar um controle paralelo. É um controle adicional, que deve aparecer no planejamento geral do projeto.

Riscos tendem a serem discutidos em níveis hierárquicos superiores na organização, um problema, visto que pode restringir a qualidade / quantidade de informações necessárias podendo levar a uma errada tomada de decisão. Gerentes precisam encorajar a comunicação aberta e destemida, garantindo que todos tenham um completo entendimento sobre os riscos do projeto.

A gerência de riscos está intimamente relacionada com a tomada de decisões perante situações de incerteza. A definição de riscos deve ser feita de forma clara, a fim de eliminar incertezas, interpretações errôneas ou várias interpretações da mesma situação. Definições claras de riscos ajudam a equipe a:

• Garantir que todos os membros tenham o mesmo entendimento sobre o risco;
• Entender a causa do risco e a relação com os problemas que podem ocorrer;
• Prover uma base quantitativa para planejar (mensurar) o esforço em uma análise formal;
• Reforçar que Stakeholders e Sponsor confiam na habilidade da equipe gerenciar riscos;

Gerenciamento de riscos

Durante o planejamento do projeto, o time deve desenvolver e documentar o gerenciamento de riscos no contexto do projeto. Nesta fase algumas questões podem ser feitas:

• Como o processo de gerenciamento de risco pode ser implementado?
• Quais são as etapas neste processo?
• Quais são as atividades, papéis, responsabilidades e resultados de cada etapa?
• Quem irá realizar atividades de risco?
• Quais são os requisitos de conhecimento?
• Existe algum treinamento adicional?
• Como o gerenciamento de riscos no projeto diz respeito a esforços de nível empresarial?
• Que tipos de ferramentas ou métodos serão utilizados?
• Que definições são usadas para classificar e estimar o risco?
• Como os riscos serão priorizados?
• Como os planos de contingência serão criados e executados?
• Como as atividades de controle de risco serão integradas ao plano geral do projeto?
• Que atividades os membros da equipe farão para monitorar o risco?
• Como o status do risco será acompanhado pelos participantes do projeto?
• Como o progresso será monitorado?
• Que tipo de infraestrutura será usada (bases de dados, ferramentas, repositórios) para apoiar o processo de gestão de riscos?
• Quais são os riscos do gerenciamento de riscos?
• Que recursos estão disponíveis para gestão de riscos?
• Quais são as datas importantes no planejamento para implementar o gerenciamento de risco?
• Quem é o sponsor (patrocinador) e quem são os stakeholders?

Visão geral dos processos de gerenciamento de risco

O gerenciamento de riscos consiste em uma avaliação contínua integrada ao processo de tomada de decisão, além do monitoramento e gerenciamento constante até que os riscos sejam eliminados ou se tornem problemas que possam ser resolvidos. A **Figura 2** ilustra os passos desse processo. Note que estes passos não são executados em uma sequência específica, mas sim lógica (Microsoft, 2002).

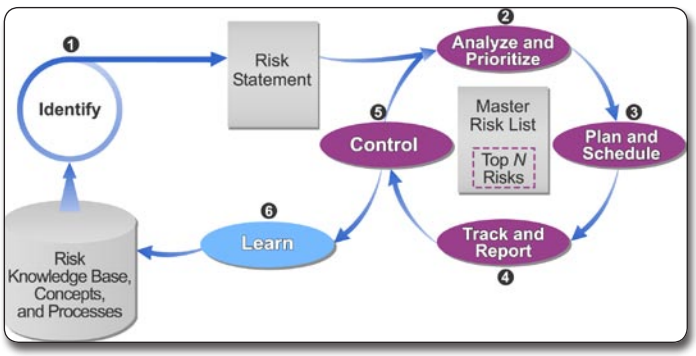


Figura 2. Processo de gerenciamento de riscos (Fonte - Microsoft, 2002)

1 - Identificar: Riscos devem ser identificados o mais cedo possível e frequentemente avaliados durante todo o ciclo de vida do projeto. Serve para que os indivíduos do time saibam claramente e com antecedência os potenciais riscos do projeto;
2 - Analisar: Transforma as estimativas e demais dados sobre os riscos identificados em uma forma que possam ser usados como fonte para tomadas de decisão e priorização; A priorização de riscos norteia o time a investir mais esforço e recursos nos riscos mais importantes (críticos);
3 - Planejar: Usa a informação analisada sobre riscos para elaborar estratégias, planos e ações. Nesta etapa, os riscos são aprovados e as atividades referentes ao seu gerenciamento são incluídas no cronograma geral do projeto;
4 - Rastrear: Avalia continuamente o status de um risco e o progresso dos planos de ação. Inclui o monitoramento da probabilidade e impacto, que podem consequentemente alterar outros pontos importantes, como cronograma, alocação de recursos e features do projeto;
5 - Controlar: Consiste no processo de executar um plano de ação. Pode incluir a abertura de solicitações de mudança no cronograma, alocação de recursos e features do projeto;
6 - Aprender: Formaliza as lições aprendidas de maneira que possam ser reutilizadas em outros projetos, como experiência (lição).

Identificando riscos

O primeiro passo no gerenciamento de riscos é a identificação. Essa identificação deve ser clara, sem equívocos, para que então o time passe para a análise e planejamento. Especial atenção deve ser dada à atividade de aprendizado, a fim de identificar potenciais falhas no projeto. As fontes básicas para identificação de riscos são provindas do conhecimento geral, bem como conhecimentos específicos na natureza do negócio, fatores técnicos, organizacionais e ambientais. No início do projeto é importante promover sessões de brainstorm, workshops, discussões em grupo, a fim de coletar o máximo de informações possíveis.

Muitas equipes promovem brainstorms para identificação de potenciais riscos somente nas fases iniciais do projeto, não repetindo essa atividade durante o ciclo de vida do projeto. Esta atividade deve ter uma periodicidade repetitiva, como diária, semanal ou mensal, ou de acordo com um milestone do projeto.

As classificações de risco (categorias), algumas vezes chamadas de taxonomias de risco, têm múltiplos propósitos para a equipe de desenvolvimento. Durante a identificação, pode ajudar a verificar as diferentes áreas do projeto que podem ser afetadas pelo risco. Durante brainstorms, podem ajudar a reduzir a complexidade de riscos, agrupando os riscos semelhantes. Podem fornecer uma terminologia comum a ser utilizada por todos os membros do time durante o projeto. E finalmente, as classificações servem para as empresas criarem uma base de conhecimento reutilizável sobre riscos.

A lista a seguir descreve as categorias de risco:

- **Pessoas**
  - Clientes
  - Usuários finais
  - Sponsors
  - Stakeholders
  - Pessoal
  - Organização
  - Conhecimento
  - Político
  - Moral
- **Processos**
  - Missão e objetivos
  - Tomada de decisão
  - Características do projeto
  - Orçamento, custo e cronograma
  - Requisitos
  - Projeto
  - Construção
  - Testes
- **Tecnológicos**
  - Segurança
  - Ambientes de desenvolvimento e teste
  - Ferramentas
  - Distribuição (Deploy)
  - Suporte
  - Ambiente operacional
  - Disponibilidade
- **Ambientais**
  - Legal
  - Regulamentação
  - Competição
  - Econômico
  - Tecnológico
  - Negócio

Uma declaração de risco é uma expressão em linguagem natural que relaciona casualmente um estado real ou existente do projeto com um estado eventual (Microsoft, 2002). A primeira parte da expressão é chamada de *condição* e a segunda de *consequência*, ligados por um termo como "portanto" ou "como resultado" que indique uma incerteza (menos de 100%). Esta forma de compor a expressão tem a vantagem de definir as consequências do risco com as condições observáveis (potencialmente controláveis). O resultado desta fase é uma lista de riscos, expressa de forma clara, não ambígua, usada como entrada para a próxima fase, a análise.

A **Tabela 1** mostra um exemplo de uma "lista master" de riscos.

Analisando e priorizando riscos

A análise de risco é a segunda etapa no gerenciamento de riscos, vindo após a etapa de identificação. Consiste na conversão de dados em formatos que tornem fácil a tomada de decisões. A priorização, atividade feita nesta etapa, garante que os membros do time empreguem esforços primeiro nos riscos mais importantes. Uma lista "top" (10 itens) pode ser definida neste sentido, e pode ser modificada conforme o projeto evolui. Riscos podem ser também desativados quando o time decide que não merecem mais atenção.

A taxa de probabilidade de risco deve ser um valor em percentagem, maior que zero e menor que 100. Não deve ser igual à zero para não representar que um risco não é uma ameaça. Não deve ser 100% pois um risco em sua essência não é um evento certo, mas uma eventualidade, incerta, caso contrário representaria um problema já conhecido. Como esse valor é uma estimativa, seu

Causa Raiz	Condição	Consequência	Efeito
Equipe inadequada	As funções de desenvolvedor e tester foram combinadas	Nós venderemos com mais bugs	Reduzirá a satisfação do cliente
Mudança de tecnologia	Nossos programadores estão trabalhando com uma nova linguagem de programação	Tempo de desenvolvimento será longo	Chegaremos ao mercado com atraso frente aos concorrentes
Organização	O time de desenvolvimento está dividido entre São Paulo e Belo Horizonte	Comunicação entre o time será prejudicada	Atraso na entrega do produto devido a retrabalho

Tabela 1. "Lista Master" de riscos durante a fase de identificação (Microsoft, 2002)

valor pode não ser eficiente. Nesse caso, a experiência da equipe em projetos anteriores ou análise de relatórios de mercado podem ajudar. A probabilidade, ao invés de um valor numérico, pode também ser expressa em linguagem simples de escala (baixa, média, alta) ou ainda mais natural (altamente improvável, pouco provável, quase certa).

O impacto de um risco é uma estimativa de gravidade dos efeitos colaterais ou magnitude de perda. Pode ser expressa diretamente em termos financeiros ou uma escala de medida subjetiva (por exemplo, escala de 1 a 10 ou baixo, médio, alto e crítico).

A medida de exposição ao risco mede a ameaça total do mesmo ao projeto. A taxa de exposição é um valor numérico simples calculado a partir da probabilidade x impacto. Métodos adicionais quantitativos podem ser usados de acordo com outras prioridades que não só probabilidade x impacto, como por exemplo, time, stakeholders, infraestrutura (processo e ferramentas). A **Tabela 2** mostra um exemplo que ajuda a identificar o grau de exposição de um risco.

A lista de riscos mestre (master) é uma compilação das informações obtidas com as análises de impacto e probabilidade, já com a priorização. A lista serve como base para as demais etapas do gerenciamento de riscos e para correta tomada de decisão sobre esforços de priorização, definição de ações críticas e dependências. A **Tabela 3** mostra a lista de riscos mestre com prioridade e definição de impacto, probabilidade e exposição.

O formulário de riscos ajuda na análise de riscos de forma individual, fornecendo informações adicionais à lista de riscos mestre. Alguns campos contidos neste formulário podem incluir Identificação do risco, fonte, condição, consequência, probabilidade, classificação de impacto, impacto, exposição, contexto e riscos relacionados. Um exemplo pode ser visto na **Tabela 4**.

	Impacto		
Probabilidade	**Baixo = 1**	**Médio = 2**	**Alto = 3**
Alta = 3	3	6	9
Média =2	2	4	6
Baixa = 1	1	2	3

Tabela 2. Riscos - Probabilidade x Impacto (Microsoft, 2002)

Planejamento e cronograma de riscos

Planejamento envolve o desenvolvimento de estratégias e ações detalhadas para os riscos, atividades estas integradas ao cronograma do projeto. Ao desenvolver o planejamento para redução de exposição ao risco, algumas atividades podem ser consideradas:

- Concentrar nos riscos com maior exposição;
- Tratar a condição para reduzir a probabilidade;
- Olhar para as causas em vez de sintomas;
- Lidar com as consequências para minimizar o impacto;
- Determinar a causa raiz, então olhar para situações semelhantes em outras áreas que podem surgir a partir da mesma causa;
- Estar ciente de dependências e interações entre os riscos.
- Várias abordagens podem ser consideradas para redução de riscos:
- Alocar recursos adicionais;
- Transferir a responsabilidade para indivíduos que tenham a capacidade de intervir;
- Durante o planejamento de ações de risco, a equipe deve considerar as seguintes alternativas quando formular planos de ação:

Prioridade	Condição	Consequência	Probabilidade	Impacto	Exposição
1	Cronograma longo	Fundos insuficientes ao final do ano	80%	3	2.4
2	Sem padrões de codificação para a nova linguagem	Mais bugs	45%	2	0.9
3	Sem especificação de requisitos	Algumas features não implementadas	30%	2	0.6

Tabela 3. Riscos priorizados e mensurados (Microsoft, 2002)

Item	Descrição
Identificação	Performance
Fonte	Abordagem Web state-full
Condição	A tecnologia usada para desenvolvimento Web não tem cases de sucesso comprovados, em ambientes reais
Consequência	A aplicação pode apresentar problemas de performance
Probabilidade	80%
Classificação de impacto	Alta
Impacto	3
Exposição	2.4
Contexto	Tecnologias que não possuem casos de sucesso são um forte indicativo que a mesma não se aplica a ambientes de produção, apenas a elaboração de protótipos
Riscos relacionados	Escalabilidade dos servidores Uso de memória

Tabela 4. Formulário de riscos

• Pesquisar - Não sabemos o suficiente sobre esse risco? Não precisamos estudar o risco para adquirir informações adicionais e determinar melhor suas características, antes que nós possamos decidir que ação tomar?
• Aceitar - Podemos viver com as consequências se o risco, caso fosse realmente ocorrer? Podemos aceitar o risco e não tomar nenhuma outra ação?
• Evitar - Podemos evitar o risco alterando o escopo?
• Transferir - Podemos evitar o risco transferindo-o para outro projeto, equipe, organização ou indivíduo?
• Mitigação – A equipe pode fazer algo para reduzir a probabilidade ou impacto do risco?
• Contingência - Pode o impacto ser reduzido através de um plano de reação?

A equipe pode incluir informação adicional sobre os planos em documentos que incluem: identificador, declaração de condição, estratégia de mitigação, métricas da estratégia de mitigação, itens de ação, estratégia de contingência, critérios de ação, métricas da estratégia de contingência, responsabilidades. A **Tabela 5** mostra um exemplo deste formulário.

Rastreamento e relatório de riscos

O rastreamento de risco é essencial para implementar os planos de ação efetivamente. Garante que as tarefas atribuídas na implementação das medidas preventivas e planos de contingência estão de acordo com o planejamento, cronograma e recursos. A principal atividade realizada pelo time é monitorar se as atividades desencadeadas pelos planos de ação estão funcionando. Os documentos que são considerados no início desta atividade são os formulários de ação que contêm os planos de mitigação e contingência.

Durante a atividade de monitoramento de riscos o time executa ações dos planos de mitigação. O progresso destes itens de ação e mudanças relevantes nos critérios de disparo são capturados e usados para criar possíveis relatórios sobre os status de cada risco. Alguns exemplos de métricas de projeto que podem ser continuamente monitoradas: bugs em aberto, média de horas logadas por desenvolvedores por semana, número de revisões de requisições (modificados) por semana etc.

Os relatórios de riscos acontecem em dois níveis. Para a equipe interna, o relatório de status de riscos pode considerar situações como: risco resolvido, completando o plano de ação; as ações estão de acordo com o planejamento, neste caso tudo continua como planejado; algumas ações apresentam variações conforme o plano original, medidas de ajustes devem ser definidas e implementadas; a situação de alguns riscos mudou de forma significante, o que envolve a reanálise dos riscos e replanejamento das atividades. Para os stakeholders externos, o time deve expor somente os riscos "top" de forma sumarizada. Como a equipe realiza ações para gerenciar riscos, a exposição ao risco total para o projeto deve começar a se aproximar dos níveis aceitáveis.

Como saída, esta fase origina um relatório de status de risco que informa o progresso dos planos de mitigação. Informações relevantes neste relatório podem incluir nome do risco, classificação, probabilidade, impacto e exposição na identificação, impacto e exposição atuais, nível do risco, sumário dos planos de contingência e mitigação, preparação dos planos de contingência, ações planejadas e proprietário do risco. Já o relatório executivo externo para stakeholders inclui as seguintes informações: nome do projeto, nível do risco por área do projeto, tendência, sumários das atividades dos planos de mitigação e contingência.

Controle de risco

As ações corretivas são iniciadas baseadas nas informações obtidas na fase de rastreamento. Os resultados e as lições aprendidas com a execução de planos de contingência são então incorporados a um relatório de status de planos de contingência para que as informações passem a fazer parte do projeto como base de conhecimento sobre riscos. É importante capturar o máximo de informação possível sobre os problemas quando eles ocorrem ou sobre um plano de contingência quando ele é invocado a fim de determinar a eficácia desse plano ou estratégia no controle do risco.

Item	Descrição
Identificação	Performance
Declaração de condição	A tecnologia usada para desenvolvimento Web não tem cases de sucesso comprovados, em ambientes reais, portanto a aplicação pode apresentar problemas de performance.
Estratégia de mitigação	Utilizar o mínimo de objetos de sessão que possam consumir memória exaustiva, como grandes conjuntos de dados.
Métricas da estratégia de mitigação	Aplicação por sessão no servidor pode consumir no máximo 500KB (por usuário).
Itens de ação	Uso do padrão flyweight para reduzir consumo de memória de objetos; compartilhamento de dados em cache no servidor comuns a todos usuários; Data conclusão 1 Junho 2012, por Guinther Pauli.
Estratégia de contingência	Aumento do número de servidores da Web Farm; Aumento da capacidade de memória e processamento de cada servidor; Considerar mudança de tecnologia para state-less.
Critérios para disparo da estratégia de contingência	Servidor apresentar resposta de mais de 2 segundos para uma carga de 20 usuários simultâneos.
Métricas da estratégia de contingência	Tempo de resposta inferior a 1 segundo para cada usuário com um único servidor em configuração padrão (carga min. 100 usuários)
Responsabilidades	Guinther Pauli, desenvolvedor

Tabela 5. Formulário de riscos

O objetivo da etapa de controle de risco é a execução bem sucedida dos planos de contingência. As entradas para a etapa de controle de risco são os formulários de risco e ações que detalham as atividades a serem realizadas por membros da equipe de projeto, além de relatórios de status de risco com os critérios para disparo da estratégia de contingência, indicando que um valor tenha sido excedido. As atividades do controle de risco devem utilizar os processos de gerenciamento de projeto, para monitorar e avaliar o progresso ao longo de um planejamento de ação. É importante manter uma contínua identificação de riscos para detectar riscos secundários que possam surgir. As saídas para esta fase são o documento padrão que reporta o status atual de cada risco e o progresso dos planos de ação. É importante para o time sumarizar as lições aprendidas, por exemplo, o que funcionou e o que não funcionou. Mudanças no status de riscos podem sugerir mudanças no calendário, recursos ou features do projeto, principalmente se originarem processos formais de solicitações de mudança.

Aprendendo com riscos

O objetivo desta etapa é sumarizar os resultados e experiências aprendidas com o gerenciamento de risco no projeto, agregando valor para a organização, aumentando a maturidade da equipe. O aprendizado de riscos deve ser um processo contínuo e pode iniciar a qualquer momento. Esta atividade tem três objetivos principais: garantir a qualidade sobre as atividades de gerenciamento de riscos; capturar lições aprendidas, especialmente sobre identificação e sucesso de planos de contingência, como benefício para outros times (isso irá contribuir para a construção de uma base de conhecimento comprovada); melhorar o processo de gerenciamento de riscos capturando feedback do time;

Se a equipe encontra um problema que não tinha sido identificado anteriormente como um risco, deve então verificar se algum indicador poderia ter ajudado a prever o risco. Pode ser que as listas de riscos existentes precisem ser atualizadas para ajudar em futuras identificações de riscos. Alternativamente, a equipe pode ter identificado um risco novo no projeto, que deve ser adicionado à base de conhecimento já existente. Outro ponto chave no aprendizado para capturar experiências são estratégias bem sucedidas para mitigar riscos. Neste ponto criar uma classificação dos resultados pode ajudar a futura identificação de riscos.

Conclusão

A disciplina de gerenciamento de riscos é vital para o sucesso de um projeto de software. Saber antecipar riscos pode significar oportunidade para a empresa. Controle de riscos é tão importante quanto o próprio controle de requisitos. Um risco, quando identificado prematuramente, não irá causar efeitos indesejáveis em um projeto quando se tornar um evento, sua equipe estará preparada para lidar com ele. Muitos projetos fracassam totalmente, simplesmente por que um único risco não foi identificado quando deveria.

Guinther Pauli
guintherpauli@gmail.com
Arquiteto de Software certificado pela Microsoft, Professor de Pós-Graduação, Membro Editorial das Revistas .NET Magazine Brasil e ClubeDelphi - Certificado oficial MCP, MCAD, MCSD.NET, MCTS, MCPD (ASP.NET C# - 4.0) - Certificado Delphi: 3, 5, 6, 7 Advanced, 2005, 2006, Delphi para Web, Delphi para Linux. Foi arquiteto e desenvolvedor do sistema de doações on-line do projeto Criança Esperança da Rede Globo / Unesco. Beta Test de várias versões do Delphi, palestrante em 5 edições da Borland Conference, maior evento de desenvolvedores do país. Ministra treinamentos corporativos para empresas e universidades, já tendo treinado mais de 5 mil profissionais em todo Brasil e exterior, incluindo grandes empresas como SONY, UNICAMP e KBR. Autor de mais de 800 vídeo-aulas de ensino a distância, publicou mais de 130 artigos, além do livro "Delphi - Programação para Banco de Dados e Web". Fez a cobertura de grandes eventos no exterior, como o TechEd da Microsoft e a BorCon da Borland americana. Desenvolve em Delphi há 17 anos e 10 anos em .NET / C#.

Microsoft Solutions Framework - MSF Risk Management Discipline.
http://www.microsoft.com/msf

@GuintherPauli
Facebook.com/GuintherPauli
http://www.gpauli.com

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Seu **conhecimento vale muito** na DevMedia

Publique artigos e vídeos gratuitamente no site e participe do programa **PayContent.**

Sua participação pode ser premiada: quanto mais você postar, mais chances de ganhar.

Para saber mais acesse:
www.devmedia.com.br/paycontent

DevMedia

TENHA ACESSO ILIMITADO A TODO CONTEÚDO DO SITE DEVMEDIA

SEJA UM ASSINANTE MVP!

Tenha acesso garantido a qualquer tema que você possa necessitar!

VIDEOS -AULAS

O assinante MVP acessa todas as vídeo-aulas do site DevMedia. Nenhum outro produto dá essa possibilidade. São mais de **7.000 video-aulas.**

CURSOS

Nosso site possui atualmente **+ de 250 cursos online** com a qualidade que você já conhece e confia. Com o MVP você terá acesso a todos eles.

REVISTAS

Assinante MVP acessa todas as nossas revistas na versão digital, incluindo todas as edições anteriores. Quando pintar aquela dúvida você terá mais de **3.500 artigos para consultar!**

ASSINATURA MVP MOST VALUABLE PROFESSIONAL

A assinatura MVP te dá a oportunidade de aumentar seu conhecimento em diversos assuntos, com qualidade e economia.

Invista na sua carreira se torne um profissional ainda mais completo!

POR APENAS R$59,90 POR MÊS

DevMedia

Acesse: www.devmedia.com.br/mvp